APPLE 軟體硬體
應用程式

廖 敏 芳譯



博文出版社出版

APPLE 軟體硬體
應用程式

廖敏芳 譯

博 文 出 版 社 出 版

APPLE 軟體、硬體應用程式

編譯者：**廖　敏　芳**
出版者：博　文　出　版　社
發行者：博　文　出　版　社
九龍大連排道872號
印刷者：鴻　文　印　刷　廠
香港柴灣工廠大厦五樓

定價：H. K. $ 25.00

目　　錄

簡　介 ………………………………………………… 1

1　機器語言輔助程式 ………………………………… 3

BREAKER：一個 APPLE II 的除錯輔助程式
—— *Rick Auricchio* ………………………… 7

APPLE II PLUS 的 STEP 和 TRACE
—— *Craig Peterson* ………………………… 22

TRACER：APPLE II 的一個除錯工具
—— *R. Kovacs* ………………………… 30

APPLE 整數 BASIC 副常規 PACK 和 LOAD
—— *Richard F. Suitor* ………………………… 38

MEAN-14：APPLE II 的虛假浮點處理機
—— *R. M. Mottola* ………………………… 50

2　加强輸入/輸出功能 ………………………………… 65

螢幕的寫入 / 存檔常規
—— *B. E. Baxter* ………………………… 69

雙向自動捲頁

—— *Roger Wagner* ······ 74

一頁一頁的列印APPLE II 整數BASIC程式

—— *Dave Partyka* ······ 82

APPLE印表機的編頁輸出

—— *Gary Little* ······ 88

十六進位印表機

—— *Leroy Moyer* ······ 94

3 執行時用的公用程式 ······ 99

APPLE II 的共用變數

—— *Robert F. Zant* ······ 103

APPLESOFT的PRINT USING

—— *Gary A. Morris* ······ 109

搜尋字串陣列

—— *Gary B. Little* ······ 117

APPLESOFT與矩陣

—— *Cornelis Bongers* ······ 123

AMPER-SORT

—— *Alan G. Hill* ······ 137

APPLE II 的追踪列印公用程式

—— *Alan G. Hill* ······ 155

4 圖形與遊戲 ······ 165

一個多用途的高解析度函數繪圖程式
——— *David P. Allen* ……………………… 168
縮減 APPLE II 高解析度照片所佔的記憶體空間
——— *Bob Bishop* ……………………… 173
一個使APPLE的LIFE遊戲更有效率的程式
——— *Gregory L. Tibbetts* ……………………… 190
Applayer 音樂解析器
——— *Richard F. Suitor* ……………………… 202
改進星際戰爭的音響效果
——— *William M. Shryock, Jr.* ……………………… 216
Galacti- Cube
——— *Bob Bishop* ……………………… 217

硬　體 ……………………… 223

APPLE II 的彩色槍 (COLOR GUN)
——— *Neil D. Lipson* ……………………… 226
APPLE II 的卡帶作業系統
——— *Robert A. Stein, Jr.* ……………………… 231
利用Micromodem II 傳送 BASIC 和機器語言程式
——— *George J. Dombrowski, Jr.* ……………………… 240
APPLE II 的數位溫度計
——— *Carl Kershner* ……………………… 246
在APPLE II 上使用KIM和SYM 格式的卡帶

——— *Steven M. Welch* ……………………251

6 參考資料 ……………………261

從整數BASIC截取DOS錯誤訊號

——— *Andy Hertzfeld* ……………………264

APPLESOFT的浮點常規

——— *R. M. Mottola* ……………………269

如何使用傳送口（hooks）

——— *Richard Williams* ……………………278

BROWN和WHITE與所有的彩色

——— *Richard F. Suitor* ……………………288

簡 介

MICRO 雜誌爲APPLE 使用者提供軟體方面的服務已經四年了。在此書中，我們把過去幾年來一些比較有名的程式與文章，集合在一起，推荐給APPLE 使用者。

雖然MICRO 每個月都有新的程式與文章出版，而一些較舊的MICRO 文章仍然是最好用的。在這麼多的文章中，我們選擇了一些較好的文章，即最具代表性的文章，集合在此書中。

MICRO 雜誌爲了迎合一些認眞的APPLE 使用者的需要，大部份的文章都附有程式——這些程式說明一些有用的技巧，或是執行一些較複雜的工作。在此書中，我們在每篇文章後也附有一個或一個以上的程式。這些程式不管是整數BASIC 、 APPLESOFT 或機器語言程式，都已經被輸入在磁碟中，而且已經測試過。

此書另外還包括了一片 13- 磁扇（ sector ）的磁碟。選擇這種 13- 磁扇磁碟的理由是，它能普遍性的適合 DOS 3.2格式，並且很多的系統均沒有DOS 3.3 。而且經由APPLE的 MUFFIN 程式可以很容易的將此格式的磁碟轉換成 DOS 3.3 格式——磁碟內的程式仍可正常的作業。

我們希望我們收集這些有名的文章與程式（在磁碟上）能夠讓使用者做一些試驗，並探討程式所使用的一些技巧，使您對您的APPLE更進一步的了解。

1　機器語言輔助程式

簡　介

BREAKER: 一個 APPLE II 的除錯輔助程式

——— Rick Auricchio

APPLE II PLUS 的 STEP 和 TRACE

——— Craig Peterson

TRACER: APPLE II 的一個除錯工具

——— R. Kovacs.

APPLE 整數 BASIC 副常規 PACK 和 LOAD

——— Richard F. Suitor

MEAN-14: APPLE II 的虛假浮點處理機

——— R. M. Mottola

第一部份簡介

本章將介紹一些公用程式（ Utility Program ），使用者可利用這些程式使得在以機器語言設計程式時，較容易且較省時。這些公用程式大部份都能連在一起使用，所以在讀完本章以後，熱愛機器語言的程式設計師將擁有非常多的工具。由 Richard Auricchio 所寫的“ BREAKER ：一個 APPLE II 的除錯輔助程式”，它提供使用者在程式中設立中斷點（ breakpoints ）的功能，是一個威力很大的除錯程式。由 Craig Peterson 所寫的“ APPLE II PLUS 的 STEP 和 TRACE ”，它提供 STEP 和 TRACE 的功能給自動啓動監督系統 ROM ，而這二項功能本來只存在不連續的舊監督系統 ROM 中。由 R. Kovacs 所寫的“ TRACER ：APPLE II 的一個除錯工具”，將使您的監督系統或 Peterson 的程式增加 STEP 或 TRACE 的功能。以上三個常規主要是除錯用的。

Richard Suitor 的“ APPLE 整數 BASIC 副常規 PACK 和 LOAD ”，提供一個非常容易的方法“將機器語

言程式合併在整數 BASIC 驅動程式中”。這個程序將會使得在磁碟或磁帶上儲存程式的動作簡化。最後，R. M. Mottola 的“MEAN 14：APPLE II 的虛假浮點處理機”，它提供一種機器語言，它和 APPLESOFT 一樣具有處理浮點的功能。

BREAKER:

一個APPLE II的除錯輔助程式

作者： Rick Auricchio

發展機器語言程式通常可利用中斷點（ break-points）來加快完成。然而 APPLE II 並沒有提供設立中斷點的能力，此程式將提供這種功能。在機器語言程式中，任何時間任何地方均可加入或消除多個中斷點。

當對一個組合語言程式進行除錯時，程式師所能用的最簡單的工具就是設立中斷點。最基本的型式就是，中斷點的設立使得硬體方面能停止 CPU 於一特定的位址上：用 “ deluxe ”方式可利用Read/write 以及 Sync （指令尋取）線，允許停在一特定的指令，載入一位元組，或者將一個位元組存入記憶體。因爲軟體較硬體容易，利用 6502 CPU 中的 BRK 運算碼，我們可以用軟體來做。

一個中斷點，事實上就是在程式中加入一個 BRK 的運算碼。當使用者的程式執行到 BRK 時，控制權將會轉移到監督系統（經由 IRQ 向量 $FFFE/FFFF ）。在APPLE中，監督系統將會保存程式的狀況和暫存器，並印出暫存器

的值，並把控制權還給鍵盤。然而，困難的是我們希望再繼續執行程式：必須將 BRK 移開，以原來的指令取代，而且原來的暫存器值也要存回去，以便能繼續執行。然而，如果我們只是以原來的運算碼取代， BRK 就不見了，而程式還會再執行那位址，就不會有中斷的情形發生了。

此問題的答案就是 BREAKER ：一個用來管理中斷點的軟體常規。此除錯程式所做的事很簡單：它管理中斷點的加入與消除，並且會正確的繼續執行程式。當程式繼續執行時，原來的指令將會自動的執行。

這是不是很奇妙

不，這不奇妙，只是讓計算機記住中斷點的位址而已！如果除錯程式知道中斷點的位址，則它也會知道原來的指令在那裡。擁有這些資訊，我們便可容易的管理中斷點了。底下是除錯程式工作的情形。

開始時，BREAKER經由Control-Y和COUT和APPLE的監督系統連接。Control-Y 是用來處理除錯命令，而 COUT 的作用是，當 BRK 發生時，它用來從監督系統中"偸取控制權"。

中斷點的資訊是存在表格中： LOCTAB 是一個含有二個位元組位址的表格——它含有中斷點設立的位址。ADTAB 是一個含有一個位元組位址的表格：它用來尋找一

個中斷點表的起始點（Break Table Entry（BTE））。BTE有12個位元組（只有前9個用到），且包含下列各項:

* 原來使用者程式的指令；
* 跳回使用者程式；
* 對相關的分歧目標跳回。

當加入一個中斷點時，我們必須正確的設立 BTE，並且將程式中的中斷位址放入 LOCTAB。這兒允許8個中斷點存在，所以我們有一個 16 位元組的 LOCTAB、8位元組的 ADTAB、以及 96 位元組的 BTE。

當加入一個中斷點時，原來的指令便被拷貝到 BTE 的前3個位元組中，並且如果它是1個或2個位元組的指令時，要配合NOP指令（$EA）。一個 BRK 的運算碼將取代原來指令的運算碼（其它的指令均不改變）。BTE 的下3個位元組將包含一個 JMP 指令，以便跳到程式的下一個指令。

如果原來的指令是一個相對分歧指令，則必須另外考慮一點：假如我們將此相對分歧指令移到 BTE 中，則它如何正確的跳回？解決的方法就是在 BTE 中設立另外一個JMP指令——跳到分歧的目標，即將分歧指令的PC 加上2，再加上位移（offset）。這個 JMP 的絕對位址將放在BTE的7～9位元組中。其中位移是由原來指令拷貝來的，其值變成$04，所以才會跳到 BTE 的第二個 JMP 指令，此

JMP 指令將會跳到正確的位置。

呼叫監督系統中的“ INSDS2 ”常規，將會傳回其指令的長度與種類。運算碼由 AC 提供，且此常規適當的設定 LENGTH 和 FORMAT 之值。

將一個中斷點除去只需要存回原來的運算碼，並且清除 LOCTAB ，以便使此中斷點的 BTE 能再被使用。

中斷點的顯示程式將會印出中斷點的位址，接著是對應此中斷點的 BTE 位址（ BTE的位址是非常有用的——其重要性在後面說明）。

當執行到一個中斷點時，APPLE 的監督系統將會取得控制權。監督系統會嗶一聲，並印出程式中暫存器的值。在印暫存器值的時候， BREAKER 將經由 COUT 取得控制權。（記住，我們控制每一個印出的字——但只有在印暫存器的值時最重要。即當我們在一個中斷點時。）當取得控制權的時候，BREAKER 將取得 PC 值並存起來（因爲 BRK 指令的關係，所以要減 2 ）。如果在此 PC 不存在一個中斷點（我們可以掃瞄 LOCTAB ），則繼續執行監督系統。如果存在一個中斷點，則將 BTE 的位址當作“連續 PC ”。換句話說，當中斷之後，若要繼續執行程式就要跳到 BTE ；則原來的指令將會被執行，且會分歧到程式的剩餘部份。

BREAKER使用法

第一件事就是將 BREAKER 載入記憶體中。啓動時必須從起始位址（ start address ）進入，這將從監督系統中設立一些出口點。當重新開始時，必須經由“ Y_cI ”（ Y_c 即 Control－Y ）來重新啓動，以便再設定 COUT 的出口點。一進入起始位置後，所有的中斷點都會被清除：“ Y_cI ”之後，中斷點仍然保持作用。

要加入一個中斷點，打入： aaaaY_cA 。這將會在程式中位址 aaaa 上設立一個中斷點。嗶一聲表示有錯誤；否則在 aaaa 位址上就有一個中斷點。若要清除一個中斷點，則打入： aaaaY_cR 。這將會把 aaaa 位址上的中斷點清除，並且存回原來的運算碼。嗶一聲表示沒有中斷點。

經由監督系統的“ G ”命令來執行您的程式。當碰到一個中斷點時，暫存器的值將會印出，並且控制權會送回監督系統。此時，所有監督系統的命令都有效，包括BREAKER使用的“ Y_cA ”、“ Y_cR ”、以及“ Y_cD ”。

若要繼續執行，則打入：Y_cG。將使得 BREAKER 從 BTE 開始執行（即執行原來的指令），此時控制權將回到程式。不要使用監督系統的“ G ”命令——這行不通的，因爲監督系統對中斷點之事一點都不知道。要顯示所有的中斷點，則打入： Y_cD 。這將最多顯示 8 個中斷點，並且還有各個中斷點的 BTE 位址。

注意事項

使用 BREAKER 來對一個程式除錯時，要注意一些事項。第一，當您執行程式時， BREAKER 可能沒有啓動好。當您開始時，這不是一個問題，因爲您不能使用 Y_c 命令。但是如果在測試時按下Reset，則必須利用“ Y_cI ”重新啓動，否則對一個中斷點 BREAKER 將不會處理。如果您試 Y_cG ，則將會產生一些不可預測的事。假如您知道您打了一個中斷點，而 BREAKER 却沒有作用，則您可以令它恢復。只要打“ Y_cI ”，並且顯示中斷點（Y_cD）。再利用監督系統的“ G ”命令從中斷點的 BTE 中繼續執行程式（因爲當碰到中斷點時 BREAKER 並不存在，必須手動的從 BTE 繼續執行）。現在一切都正常。您可以利用顯示 $38 和 $39 的位置得知 BREAKER 是否有作用。如果沒有作用，它們將含有 $FOFD。

特別要注意的是，如果使用者的程式中使用到Control-Y或者COUT命令時，不能使用 BREAKER 來除錯。

BREAKER 命令摘要

命令	功能
aaaaY_cA	在 aaaa 位址上加入中斷點。若已經存在有中斷點，則不允許再加入。最多只能在程式中加

入 8 個中斷點。

Y_cD 顯示所有的中斷點。

Y_cI 在按 RESET 鍵之後重新啓動。只是設立 COUT 的出口，而沒有重新設立任何中斷點。

aaaaY_cR 把位址 aaaa 上的中斷點清除。並存回原來的運算碼。

```
0800   ;*************************
0800   ;*                       *
0800   ;*   BREAKER-DEBUGGER    *
0800   ;*    RICK AURICCHIO     *
0800   ;*                       *
0800   ;*       BREAKER         *
0800   ;*                       *
0800   ;*  COPYRIGHT (C) 1981   *
0800   ;*    MICRO INK, INC.    *
0800   ;* CHELMSFORD, MA 01824  *
0800   ;*  ALL RIGHTS RESERVED  *
0800   ;*                       *
0800   ;*************************
0800   ;
0800   ;ROUTINES TO HANDLE UP TO
0800   ;8 BREAKPOINTS, FOR USE IN
0800   ;DEBUGGING OF USER CODE...
0800   ;
0800   ;
0800   ; *** APPLE-2 MONITOR EQUATES
0800   ;
0800   FORMAT EPZ $2E
0800   LENGTH EPZ $2F
0800   A1L    EPZ $3C
0800   A1H    EPZ $3D
0800   A2L    EPZ $3E
0800   A2H    EPZ $3F
0800   A3L    EPZ $40
0800   A3H    EPZ $41
0800   ;
0800   CSWL   EPZ $36
0800   CSWH   EPZ $37
0800   ;
0800   INSDS2 EQU $F88E
0800   PRNTYX EQU $F940
0800   PRBYTE EQU $FDDA
0800   COUT   EQU $FDED
0800   RESET  EQU $FF65
```

```
0800          MON     EQU $FF69
0800          ;
0800          ; CHANGE 'LOWPAGE' TO LOCATE
0800          ; ELSEWHERE IN MEMORY. IT IS
0800          ; NOW SET FOR A 48K DOS SYSTEM.
0800          ;
0800          LOWPAG EQU $9300
0800          ;
9300                  ORG LOWPAG
9300                  OBJ $800
9300          ;
9300 4C3695   INIT    JMP INITX            ;=>INITIALIZATION ENTRY
9303          ;
9303          ; *** DATA AREAS
9303          ;
9303 00       FW1     BYT $00              ; 'FINDPC' WORK BYTE 1
9304 00       FW2     BYT $00              ; 'FINDPC' WORK BYTE 2
9305 00       PCL     BYT $00              ; 'GO' PC LO
9306 00       PCH     BYT $00              ; 'GO' PC HI
9307          ;
9307          ; SKELETON BREAK-TABLE ENTRY
9307          ;
9307 00       SKEL    BYT $00              ;SKELETON BTE
9308 EA               NOP
9309 EA               NOP                  ;NOPS FOR PADDING
930A 4CC000           JMP $00              ;JUMP BACK INLINE
930D 4C               BYT $4C              ;JUMP OPCODE FOR BRANCHES
930E          ;
930E          ; LOW ADDRESS OF BTES KEPT IN ADTAB
930E          ;
930E 26       ADTAB   BYT BTE0             ;LO ADDRESS
930F 32               BYT BTE1
9310 3E               BYT BTE2
9311 4A               BYT BTE3
9312 56               BYT BTE4
9313 62               BYT BTE5
9314 6E               BYT BTE6
9315 7A               BYT BTE7
9316          ;
9316          ; LOCTAB CONTAINS ADDRESS OF USER-PROGRAM INSTRUCTION
9316          ; WHERE WE PLACED THE BREAKPOINT IN THE FIRST PLACE
9316          ;
0826          LOCTAB DFS $10               ;SPACE FOR 16 PCH/L PAIRS
9326          ;
9326          ; BREAK-TABLE ENTRIES (BTE'S)
9326          ;
0832          BTE0    DFS $0C              ;12 BYTES RESERVED
083E          BTE1    DFS $0C
084A          BTE2    DFS $0C
0856          BTE3    DFS $0C
0862          BTE4    DFS $0C
086E          BTE5    DFS $0C
087A          BTE6    DFS $0C
0886          BTE7    DFS $0C              ;ENOUGH FOR 8 BREAKPOINTS
9386          ;
9386          ; END OF DATA AREAS
9386          ; *THE REST IS ROM-ABLE*
```

```
9386            ;
9386            ; ***********************************************
9386            ; *
9386            ; * NAME:     FINDPC
9386            ; * PURPOSE: CHECK IF PC IN FW1/FW2 MATCHES LOCTAB
9386            ; * RETURNS: CARRY SET IF YES; XREG=ADTAB INDEX 0-7
9386            ; *          CARRY CLR IF NOT; XREG=GARBAGE
9386            ; * VOLATILE:DESTROYS AC
9386            ; *
9386            ; ***********************************************
9386            ;
9386 A20F       FINDPC LDX #115           ;BYTE-INDEX TO END OF TABLE
9388 AD0493     FPC00  LDA FW2            ;GET FOR COMPARE
938B DD1693            CMP LOCTAB,X       ;A PCH MATCH?
938E D008              BNE FPC02          ;=>NO. TRY NEXT 2-BYTE ENTRY
9390 AD0393            LDA FW1            ;GET PCL NOW
9393 DD1593            CMP LOCTAB-1,X     ;A PCL MATCH?
9396 F006              BEQ FPC04          ;=>YES! WE HAVW BREAKPOINT!
9398 CA         FPC02  DEX                ;BACK UP ONE
9399 CA                DEX                ;AND ANOTHER
939A 10EC              BPL FPC00          ;=>DO ENTIRE TABLE SCAN
939C 18                CLC                ;=>DONE; SCAN FAILED
939D 60                RTS
939E            ;
939E 48         FPC04  PHA                ;HOLD AC
939F 8A                TXA                ;HALVE VALUE IN X-REG
93A0 4A                LSR                ;SINCE IT'S 2-BYTE INDEX
93A1 AA                TAX
93A2 68                PLA
93A3 38                SEC                ;SET 'SUCCESS'
93A4 60
93A5            ;
93A5            ;
93A5            ; ***********************************************
93A5            ; *
93A5            ; * NAME:     BREAK
93A5            ; * PURPOSE: HANDLE ENTRY AT BRK AND PROCESS BPOINTS
93A5            ; * NOTE:     THIS ROUTINE GETS ENTERED ON *EVERY* 'COUT
93A5            ; *           CALL-- IT KNOWS ABOUT BRK BECAUSE THE
93A5            ; * MONITOR'S REGISTERS ARE SETUP TO PRINT USER REG
93A5            ; * CONTENTS. AFTER PROCESSING IS DONE, IT RESTORESTHE
93A5            ; * MONITOR'S REGS AND RETURNS
93A5            ; *
93A5            ; ***********************************************
93A5            ;
93A5 E0FB       BREAK  CPX #$FB           ;IS XREG SET FOR EXAMINE
93A7 D027              BNE BRKXX          ;=>NO GET OUT NOW.
93A9 C9A0       BRK02  CMP #$A0           ;IS AC SETUP CORRECTLY
93AB D023              BNE BRKXX          ;=>NOPE. FALSE ALARM!
93AD A53C              LDA A1L            ;GET USER PCL
93AF 38                SEC                ; AND BACK IT UP BY
93B0 E902              SBC #$02           ; 2 BYTES SINCE BRK BUMPED
93B2 8D0393            STA FW1
93B5 A53D              LDA A1H            ;GET PCH
93B7 E900              SBC #$00           ;DO THE CARRY
93B9 8D0493            STA FW2            ; AND SAVE THAT TOO
93BC 208693            JSR FINDPC         ;A BREAKER OF OURS HERE?
93BF 900B              BCC BRK04          ;=>NOPE. WE WON'T HANDLE
```

```
93C1 BD0E93         LDA ADTAB,X         ;YES, GET BTE ADDRESS THEN
93C4 8D0593         STA PCL             ; AND SET IT AS THE 'GO'
93C7 A993           LDA /LOWPAG         ; PC FOR THE 'GO' COMMAND.
93C9 8D0693         STA PCH             ;(OUR PAGE FOR BTE'S)
93CC         ;
93CC A9AC    BRKC4  LDA #$AC            ;SET AC BACK FOR MONITOR
93CE A2FB           LDX #$FB            ;AND X-REG. TOO
93D0 4CF0FD  BRKXX  JMP $FDF0           ;=>NO. RIGHT BACK TO COUT
93D3         ;
93D3         ; **********************************
93D3         ; * PROCESS THE 'GO' COMMAND....   *
93D3         ; *      (RESUME USER EXECUTION)   *
93D3         ; * COMMAND FORMAT:(* CTRL-Y G)    *
93D3         ; **********************************
93D3         ;
93D3 AD0593  CMDGO  LDA PCL             ;GET RESUME PCL
93D6 853C           STA A1L             ; AND SETUP TO SIMULATE
93D8 AD0693         LDA PCH             ; AN 'XXXX G' COMMAND
93DB 853D           STA A1H
93DD 4CB9FE         JMP $FEB9           ;=>SAIL INTO MONITOR'S 'GO'
93E0         ;
93E0         ; **********************************
93E0         ; * WE GET CONTROL HERE ON THE     *
93E0         ; * CNTL-Y USER EXIT FROM THE      *
93E0         ; * MONITOR (ON KEY-INS).  ALL     *
93E0         ; * COMMANDS ARE SCANNED HERE;     *
93E0         ; * CONTROL WILL PASS TO THE       *
93E0         ; * APPROPRIATE ROUTINE.....       *
93E0         ; **********************************
93E0         ;
93E0 A2FF    KEYIN  LDX #$FF            ;CHAR INDEX
93E2 E8      KEYINO INX                 ;SET NEXT CHARACTER
93E3 BD0002         LDA $200,X          ;GET CHARACTER FROM BUFFER
93E6 C999           CMP #$99            ;CONTROL-Y CHARACTER?
93E8 D0F8           BNE KEYINO          ;=>NO. KEEP SCANNING
93EA E8             INX                 ;BUMP OVER CTRL-Y
93EB BD0002         LDA $200,X          ;GRAB COMMAND CHARACTER
93EE C9C7           CMP #$C7            ;IS IT 'G' (GO)?
93F0         ;
93F0         ; A BRANCH TABLE WOULD BE NEATER,
93F0         ; BUT IT WOULD TAKE UP MORE CODE
93F0         ; FOR THE FEW OPTIONS WE HAVE...
93F0         ;
93F0 F0E1           BEQ CMDGO           ;=>YES.
93F2 C9C1           CMP #$C1            ;IS IT 'A' (ADD)?
93F4 F018           BEQ CMDADD          ;=>YES.
93F6 C9C4           CMP #$C4            ;IS IT 'D' (DISPLAY)?
93F8 F00B           BEQ XXDISP          ;=>YES.
93FA C9D2           CMP #$D2            ;IS IT 'R' (REMOVE)?
93FC F00A           BEQ XXREMV          ;=>YES.
93FE C9C9           CMP #$C9            ;IS IT 'I' (INIT)?
9400 F009           BEQ XXINIT          ;=>YES.
9402 4C65FF  BADCMD JMP RESET           ;NOTHING, IGNORE IT!
9405         ;
9405 4CA894  XXDISP JMP CMDDSP          ;EXTENDED BRANCH
```

```
9408 4C0895  XXREMV JMP CMDRMV          ;EXTENDED BRANCH
940B 4C4F95  XXINIT JMP CMDINT          ;EXTENDED BRANCH
940E         ;
940E         ; ********************************
940E         ; *    PROCESS THE 'ADD' COMMAND    *
940E         ; * ADD A BREAKPOINT AT LOCATION  *
940E         ; * SPECIFIED IN COMMAND......    *
940E         ; * CMND FORMAT: (* AAAA CTRL-Y A)*
940E         ; ********************************
940E         ;
940E A000    CMDADD LDY #$00            ;CHECK OPCODE FIRST
9410 B13E           LDA (A2L),Y         ;OP AT AAAA A BRK ALREADY?
9412 F0EE           BEQ BADCMD          ;=>YES. ILLEGAL!
9414         ;
9414         ;SCAN LOCTAB FOR AN AVAILABLE BTE TO USE
9414         ;
9414 A2CF           LDX #115            ;BYTE INDEX TO LOCTAB END
9416 BD1693  ADD00  LDA LOCTAB,X        ;GET A BYTE
9419 D005           BNE ADD02           ;=>IN USE
941B BD1593         LDA LOCTAB-1,X      ;GET HI HALF
941E F006           BEQ ADD04           ;=>BOTH ZERO, USE IT!
9420 CA      ADD02  DEX                 ;MOVE BACK TO
9421 CA             DEX                 ; NEXT LOCTAB ENTRY
9422 10F2           BPL ADD00           ; AND KEEP TRYING
9424 30DC           BMI BADCMD          ;=>DONE? ALL FULL! REJECT
9426         ;
9426 A53E    ADD04  LDA A2L             ;GET AAAA VALUE
9428 9D1593         STA LOCTAB-1,X      ;SAVE LO HALF
942B 8D0B93         STA SKEL+4          ;STUFF LO ADDR INTO BTE
942E A53F           LDA A2H             ;GET AAAA VALUE
9430 9D1693         STA LOCTAB,X        ;SAVE HI HALF
9433 8D0C93         STA SKEL+5          ;STUFF HI ADDRESS INTO BTE
9436 8A             TXA                 ;GRAB INDEX FOR LOCTAB
9437 4A             LSR                 ;MAKE ADTAB INDEX
9438 AA             TAX                 ;AND STUFF BACK INTO X-REG
9439 A993           LDA /LOWPAG         ;BTE'S HI ADDRESS VALUE
943B 8541           STA A3H             ;HOLD IN WORK AREA
943D BD0E93         LDA ADTAB,X         ;GET BTE LO ADDR FROM ADTAB
9440 8540           STA A3L             ;SAVE IN WORK AREA
9442 A907           LDA #$07            ;7-BYTE MOVE FOR SKEL BTE
9444 B90793  ADD06  LDA SKEL,Y          ;GET SKEL BYTE
9447 9140           STA (A3L),Y         ;MOVE TO BTE
9449 88             DEY                 ;SET NEXT
944A 10F8           BPL ADD06           ;=> MOVE ENTIRE SKELETON
944C C8             INY
944D B13E           LDA (A2L),Y         ;GET ORIGINAL OPCODE
944F 9140           STA (A3L),Y         ; INTO BTE......
9451 208EF8         JSR INSDS2          ;INSDS2 (TO DISASSEMBLE)
9454 A900           LDA #$00            ;SET BRK OPCODE
9456 913E           STA (A2L),Y         ; OVER ORIGINAL OPCODE
9458 A52F           LDA LENGTH          ;GET INSTRUCTION LENGTH
945A 38             SEC
945B         ;
945B         ; SET UP JMP TO NEXT INST. IN THE BTE
945B         ;
945B A004           LDY #$04
945D 7140           ADC (A3L),Y         ;ADD TO PC FOR DESTINATION
945F 9140           STA (A3L),Y         ;STUFF INTO BTE
```

```
9461 C8                     INY
9462 B140                   LDA (A3L),Y       ;RUN UP THE CARRY
9464 6900                   ADC #$00          ; RIGHT HERE
9466 9140                   STA (A3L),Y       ;STUFF ADDRESS INTO JMP
9468 A52E                   LDA FORMAT        ;GET INSTRUCTION FORMAT
946A C99D                   CMP #$9D          ;IS FORMAT=BRANCH
946C F016                   BEQ ADDBRCH       ;=>YES. MORE TO DO
946E A52F                   LDA LENGTH        ;LENGTH=1?
9470 F00F                   BEQ CMDRET        ;=>YES. DONE
9472 6A                     ROR               ;LENGTH=2?
9473 B006                   BCS ADDLEN2       ;=>YES
9475 A002                   LDY #$02          ;LENGTH=3, 3RD BYTE TO BTE
9477 B13E                   LDA (A2L),Y       ;GET INST 3RD BYTE
9479 9140                   STA (A3L),Y       ;AND MOVE TO BTE
947B A001            ADDLEN LDY #$01          ;LENGTH=2, 2ND BYTE TO BTE
947D B13E                   LDA (A2L),Y       ;GET INST 2ND BYTE
947F 9140                   STA (A3L),Y       ;AND MOVE TO BTE
9481 4C69FF          CMDRET JMP MON           ;DONE, BACK TO MONITOR
9484                 ;
9484                 ; FOR BRANCHES, WE'VE GOTTA ADD A JMP FOR THE 'TRUE'
9484                 ; CONDITION (SINCE WE MOVED BRANCH OUT OF PROGRAM)
9484                 ;
9484 A001            ADDBRC LDY #$01          ;SET FOR 2ND BYTE
9486 B13E                   LDA (A2L),Y       ;GET DESTINATION OFFSET
9488 18                     CLC               ;AND ADD 2 BYTES TO
9489 6902                   ADC #$02          ;CONSTRUCT ABS ADDRESS
948B 653E                   ADC A2L           ;ADD TO SUBJECT-INST
948D 853E                   STA A2L
948F A53F                   LDA A2H           ;CARRY IT
9491 6900                   ADC #$00
9493 853F                   STA A2H
9495 EA                     NOP               ;(PLACE HOLDER WASTE HERE)
9496 A904                   LDA #$04          ;TRUE BRANCH TO +4
9498 9140                   STA (A3L),Y       ;PUT INTO NEW OFFSET
949A A007                   LDY #$07
949C A53E                   LDA A2L           ;GET JMP ADDRESS
949E 9140                   STA (A3L),Y       ;MOVE IT TO
94A0 C8                     INY               ;THE
94A1 A53F                   LDA A2H           ; BTE FOR
94A3 9140                   STA (A3L),Y       ; THE 'TRUE' BRANCH
94A5 B8                     CLV               ;SNEAKY BRANCH
94A6 50D9                   BVC CMDRET        ; TO EXIT...
94A8                 ;
94A8                 ; **********************************
94A8                 ; * DISPLAY ALL ACTIVE BRKPOINTS *
94A8                 ; *  COMMAND FMT: (* CTRL-Y D)     *
94A8                 ; **********************************
94A8                 ;
94A8 A20F            CMDDSP LDX #15           ;INDEX TO LOCTAB END
94AA BD1693          DISP00 LDA LOCTAB,X      ;GET A BYTE
94AD D00B                   BNE DISP04        ;=>IN USE
94AF BD1593                 LDA LOCTAB-1,X    ;TRY BOTH BYTES TO BE SURE
94B2 D006                   BNE DISP04        ;=>DEFINITELY IN USE
94B4 CA              DSPNXT DEX               ;SET NEXT ENTRY
94B5 CA                     DEX               ; IN LOCTAB
94B6 10F2                   BPL DISP00        ;=>MORE TO GO
94B8 30C7                   BMI CMDRET        ;=>DONE: EXIT TO MONITOR
```

```
94BA                ;
94BA A98D           DISP04 LDA #$8D          ;OUTPUT A CARRIAGE
94BC 20EDFD                JSR COUT          ; RETURN
94BF 8A                    TXA               ;GET INDEX
94C0 48                    PHA               ; SAVE IT
94C1 BC1693                LDY LOCTAB,X      ;GET SUBJECT-INST PCH
94C4 BD1593                LDA LOCTAB-1,X    ; AND ITS PCL
94C7 843B                  STY $3B           ;SET UP PCH/PCL
94C9 853A                  STA $3A
94CB AA                    TAX
94CC 2040F9                JSR PRNTYX        ;PRINT Y,X BYTES IN HEX
94CF 68                    PLA               ;RESTORE INDEX
94D0 48                    PHA
94D1 4A                    LSR               ;CONVERT TO ADTAB INDEX
94D2 AA                    TAX
94D3 A9BC                  LDA #$BC          ;'<' CHARACTER
94D5 20EDFD                JSR COUT          ;PRINT IT
94D8 A993                  LDA /LOWPAG       ;BTE HI ADDRESS
94DA 853F                  STA A2H           ;SET INDIRECT POINTER
94DC 20DAFD                JSR PRBYTE        ;PRINT HEX BYTE
94DF BD0E93                LDA ADTAB,X       ;GET BTE LOW ADDRESS
94E2 853E                  STA A2L           ;SET INDIRECT POINTER
94E4 20DAFD                JSR PRBYTE        ;PRINT BTE FULL ADDRESS
94E7 A9BE                  LDA #$BE          ;'>' CHARACTER
94E9 20EDFD                JSR COUT          ;PRINT IT
94EC                ;
94EC                ; DISSASSEMBLE THE ORIGINAL INSTRUCTION.
94EC                ; PICK UP ORIGINAL OPCODE FROM BTE,
94EC                ; ORIGINAL ADDRESDS FIELD FROM USER
94EC                ; PROGRAM LOCATION........
94EC                ;
94EC A9A0                  LDA #$A0          ;PRINT ONE SPACE HERE
94EE 20EDFD                JSR COUT
94F1 A000                  LDY #$00          ;INDEX
94F3 B13E                  LDA (A2L),Y       ;GET OPCODE FROM BTE
94F5 20DAFD                JSR PRBYTE        ;PRINT OPCODE
94F8 B13E                  LDA (A2L),Y       ;GET OPCODE FROM BTE
94FA 208EF8                JSR INSDS2        ; AND GET FORMAT/LENGTH
94FD 200495                JSR KLUGE         ;SNEAK INTO INSDSP @ F8D9
9500 68                    PLA
9501 AA                    TAX               ;RESTORE LOCTAB INDEX
9502 10B0                  BPL DSPNXT        ;=>DISPLAY THE REST
9504                ;
9504                ;
9504                ; KLUGE ENTRY INTO SUBROUTINE WHICH
9504                ; FORCES JSR PRIOR TO A PHA INSTRUCTION.
9504                ; WE HAVE TO JSR TO THIS JMP!!
9504                ;
9504 48             KLUGE  PHA               ;PUSH MNEMONIC INDEX
9505 4CD9F8                JMP $F8D9         ;CONTINUE WITH INSTDSP
9508                ;****END OF KLUGE****
9508                ;
9508                ;
9508                ; **********************************
9508                ; * REMOVE A BRKPOINT AT LOC AAAA  *
9508                ; *  COMMAND FMT: (AAAA CTL-Y RR)  *
9508                ; **********************************
9508                ;
9508 A53E           CMDRMV LDA A2L           ;GET ADDRESS LO
```

```
950A 8D0393             STA FW1           ;HOLD IT FOR FINDPC
950D A53F               LDA A2H           ;GET ADDRESS HI
950F 8D0493             STA FW2
9512 208693             JSR FINDPC        ;A BRKPOINT HERE??
9515 B003               BCS REMOV2        ;=>YES.
9517 4C65FF             JMP RESET         ;=>NO, BELL FOR YOU!
951A              ;
951A BD0E93       REMOV2 LDA ADTAB,X      ;GET THE LOCTAB ENTRY
951D 854C               STA A3L           ;HOLD IT
951F 8A                 TXA               ;NOW CREATE LOCTAB INDEX
9520 0A                 ASL
9521 AA                 TAX
9522 A900               LDA #$00          ;CLEAR OUT THE APPROPRIATE
9524 A8                 TAY               ;LOCTAB ENTRY FOR BKPT..
9525 9D1693             STA LOCTAB,X
9528 9D1793             STA LOCTAB+1,X
952B A993               LDA /LOWPAG       ;HI ADDRESS FOR BTE
952D 8541               STA A3H           ;HOLD FOR ADDRESSING
952F B140               LDA (A3L),Y       ;GET OPCODE OUT OF BTE
9531 913E               STA (A2L),Y       ; AND PULL BACK TO ORIGINAL
9533 4C69FF             JMP MON           ;=>ALL DONE.
9536              ;
9536              ;
9536              ; ******************************************
9536              ; *
9536              ; * INITIALIZATION CODE. ENTERED AT START
9536              ; * ADDR TO INITIALIZE.  IT CLEARS LOCTAB,
9536              ; * SETS UP THE CTL-Y AND COUT EXITS...
9536              ; *
9536              ; * AFTER EVERY RESET, MUST RESETUP WITH
9536              ; *          * CTL-Y 1
9536              ; *
9536              ; ******************************************
9536              ;
9536 A94C         INITX LDA #$4C          ;JMP OPCODE
9538 8DF803             STA $3F8          ;STUFF IN CTL-Y EXIT LOC
953B A993               LDA /KEYIN        ;KEYIN: HI ADDRESS
953D 8DFA03             STA $3FA          ;STUFF INTO JMP
9540 A9E0               LDA #KEYIN        ;KEYIN: LO ADDRESS
9542 8DF903             STA $3F9          ;STUFF INTO JMP ADDRESS
9545 A900               LDA #$00
9547 A20F               LDX #115          ;INDEX INTO LOCTAB END
9549 9D1693       INIT00 STA LOCTAB,X     ;CLEAR IT OUT
954C CA                 DEX               ;SO NO BREAKPOINTS
954D 10FA               BPL INIT00
954F              ;
954F              ; ENTER HERE AFTER HITTING RESET, PLEASE!
954F              ;
954F A9A5         CMDINT LDA #BREAK       ;BREAK: LO ADDRESS
9551 8536               STA CSWL          ;STUFF INTO 'COUT' EXIT HOOK
9553 A993               LDA /BREAK        ;BREAK: HI ADDRESS
9555 8537               STA CSWH          ;STUFF INTO 'COUT' EXIT HOOK
9557 4C69FF             JMP MON
                        END

***** END OF ASSEMBLY
```

```
***************************
*                         *
* SYMBOL TABLE -- V 1.5   *
*                         *
***************************

LABEL. LOC.   LABEL. LOC.   LABEL. LOC.

** ZERO PAGE VARIABLES:

FORMAT  002E   LENGTH  002F   A1L     003C A1H     003D A2L     003E A2H     003F
A3L     0040   A3H     0041   CSWL    0036 CSWH    0037

** ABSOLUTE VARABLES/LABELS

INSDS2  F88E   PRNTYX  F940
PRBYTE  FDDA   COUT    FDED   RESET   FF65 MON     FF69 LOWPAG  9300 INIT    9300
FW1     9303   FW2     9304   PCL     9305 PCH     9306 SKEL    9307 ADTAB   930E
LOCTAB  9316   BTE0    9326   BTE1    9332 BTE2    933E BTE3    934A BTE4    9356
BTE5    9362   BTE6    936E   BTE7    937A FINDPC  9386 FPC00   938E FPC02   9398
FPC04   939E   BREAK   93A5   BRK02   93A9 BRK04   93CC BRKXX   93D0 CMDGO   93D3
KEYIN   93E0   KEYIN0  93E2   BADCMD  9402 XXDISP  9405 XXREMV  9408 XXINIT  940B
CMDADD  940E   ADD00   9416   ADD02   9420 ADD04   9426 ADD06   9444 ADDLEN  947B
CMDRET  9481   ADDBRC  9484   CMDDSP  94A8 DISP00  94AA DSPNXT  94B4 DISP04  94BA
KLUGE   9504   CMDRMV  9508   REMOV2  951A INITX   9536 INIT00  9549 CMDINT  954F

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0222
```

APPLE II PLUS的STEP和TRACE

作者：Craig Peterson

如果在您新的 APPLE II PLUS 中沒有原來APPLE II的STEP/TRACE，若要恢復這些功能，此程式就是您所需要存回去的東西。

APPLE計算機的APPLE II PLUS 是一部非常好的機器。它的編輯能力比標準的 APPLE II 要好，並且游標的控制較好，以及有停止列印的功能。

進一步，APPLESOFT BASIC是一種功能很好的語言，我對它比整數 BASIC 偏愛。但是這些改進並不一定合算，在 ROM 監督系統中只有那麼多空間，而爲了加入一些新的功能必須犧牲一些原有的功能。結果，舊 APPLE II 機器語言的 STEP/TRACE 功能有時因爲記憶體空間的不足而停止。

大部份的人都不願錯過使用 STEP/TRACE 。除非您剛開始學組合語言程式，您可能不需要它們。但是如果您已經設計過很多組合語言程式，則 STEP/TRACE 是很有用的。它們允許您一步一步的執行機器語言的每一指令，並顯

示 6502 所有的暫存器的值，所以您可以發現存在程式中的錯誤，或者只是看看程式如何工作。 STEP 一次只執行一個指令，而TRACE 是連續的，不停止（除非碰到 BRK 指令）。

STEP-n-TRACE程式

APPLE II PLUS的擁有者不必擔心，STEP-n-TRACE在這裡幫助您。STEP-n-TRACE 只是在您的APPLE II PLUS的監督系統上加上STEP 和TRACE的功能。它的動作和監督系統的使用法都和原來的APPLE監督系統一樣。打入一個十六進位的位址，並接著一個或多個‘S’，將會從那位址開始一步一步的執行。打入一個十六進位的位址，並接着一個‘T’，就會從此位址開始追踪程式。

S和T比原來 APPLE TRACE 好的地方是，若要停止追踪程式只要按任何鍵即可（例如，空白鍵）。若要繼續追踪程式，只須要打入‘T’，就會從剛剛停止的位址繼續執行。或者您也可以只打入‘S’一步一步的測試。在S和T系統中，我們以反亮的‘x’爲提示符號（prompt），以便區別正常APPLE 的監督系統。S和T除了 STEP 和TRACE之外，也包括所有的監督系統命令。事實上，它也用到許多監督系統的功能來完成此工作。

如何使用此程式

要使用STEP-n-TRACE，首先要把它載入記憶體中，然後再打入‘CALL768’，或是從磁碟則打入‘BRUN’。除了STEP和TRACE之外，您還可以任意使用監督系統的命令。若要離開此程式，只要按‘RESET’即可，或者使用CTRL-C、或CTRL-B，您便會結束在BASIC中。

因爲程式存在位址$300到$3E9，和DOS的位址指標（從$3D0到$3E9）發生重疊現象。通常這對我來說並沒有問題。只要將此程式移到記憶體的其他位置即可避免此問題；但是在69、75、83、91、120、168以及169行的跳越位址也必須適當的調整。在S和T的程式中，利用到許多和APPLE監督系統相同的標籤（label），使您容易聯想到舊的監督系統在做什麼。

在此，我必須說明此STEP-n-TRACE功能和原來的APPLE監督系統犯有相同的問題，即在某些情況下，堆疊暫存器（stack register）將會顯示一些錯誤的資料。例如當JSR或RTS之後便會發生，而在執行下一指令之後便會正確。同樣的，如果程式利用TXS指令來處理堆疊，則可能會發生錯誤。最後，由於DOS的影響，如果程式追踪到I/O的傳送口（hook）（通常是$36或$37）改變時，程式的追踪將會被鎖住，因爲輸出將會有一個不正確的

跳越間接位址，且追踪將會失敗。上述的缺點並沒有多大關係。整體上來看，S 和 T 是一個很好用的公用程式。

利用S&T研究APPLESOFT

若您已讀過此節，但未實際設計任何組合語言程式來測試，底下將說明 APPLESOFT 如何工作。首先將 STEP-n-TRACE 載入記憶體中，並打入下列BASIC程式：

```
10 CALL 768 : PRINT " HELLO "
20 END
```

再來打入' RUN'，您就會聽到"嗶"一聲，以及在螢幕上出現反亮的" * "提示符號，告訴您是在 S & T 中。下一步再打入" FF58S "。從現在開始，您每打一個" S "便表示執行 APPLESOFT 的一個指令。第一個" S "將顯示" D823-4CD2D7JMP$D7D2 "在螢幕上，緊接着是暫存器的值。表示執行時返回 APPLESOFT。如果您再打入" S "或" T "，您將會看到冒號（ $3A ），PRINT 命令字（ $BA ）、雙引號（ $22 ），" HELLO " 的字元符號($48、45、4C、4C、4F)以及更多，都經由暫存器A送出，就像是 APPLESOFT 在分析您的程式一樣。

做一些研究，您將會開始了解 APPLESOFT 在做什麼

。再多一些努力，您便可實際找到副常規“ SIN ”、“ SQR ”或其他您有興趣的函數的位置。這些均可利用S & T來完成。

所以，如果您在APPLE II PLUS 上處理任何組合語言程式，S & T將會提供很大的幫助。假使您只對執行時所發生的事有興趣，則STEP - n - TRACE 也將為您服務。

（注意：此程式的修訂版，STEP-TRACE. 800，也包括在磁碟中。STEP-TRACE. 800 載入到 $ 800，並且沒有 STEP-TRACE 所提供按鍵停止的功能［見程式報表］。結果，在APPLE II PLUS 或語言卡系統中，STEP-TRACE. 800 可以和 TRACE 程式配合使用。要做此工作，首先要啓動STEP-TRACE.800，再求是 TRACER 。）

```
0800        ;*************************
0800        ;*                       *
0800        ;*      STEP-N-TRACE     *
0800        ;*    CRAIG PETERSON     *
0800        ;*                       *
0800        ;*       STEP-TRACE      *
0800        ;*                       *
0800        ;*  COPYRIGHT (C) 1981   *
0800        ;*    MICRO INK, INC.    *
0800        ;* CHELMSFORD, MA 01824  *
0800        ;*  ALL RIGHTS RESERVED  *
0800        ;*                       *
0800        ;*************************
0800        ;
0800        ;A PROGRAM TO FURNISH THE APPLE II
0800        ;PLUS WITH THE STEP AND TRACE CAPA-
0800        ;BILITIES OF THE STANDARD APPLE II.
0800        ;
0800        RTNL    EPZ $2C            ;RETURN ADDRESS LO
0800        RTNH    EPZ $2D            ;RETURN ADDRESS HI
0800        LGTH    EPZ $2F            ;LENGTH/DISPLACEMENT
0800        PRMP    EPZ $33            ;PROMPT CHARACTER
0800        YSAV    EPZ $34            ;PLACE TO SAVE Y
0800        PCL     EPZ $3A            ;PROGRAM COUNTER LO
0800        PCH     EPZ $3B            ;PROGRAM COUNTER HI
```

```
0800          XQT    EPZ $3C        ;USER INSTRUCTION
0800          STAT   EPZ $48        ;PROC STATUS REG
0800          ;
0800          ;
0800          KBRD   EQU $C000      ;KEYBOARD REGISTER
0800          INSD   EQU $F882      ;DISPLAY PRGRM CNTR
0800          DISA   EQU $F8D0      ;DISASEMBL INSTR
0800          ADJ2   EQU $F954      ;ADJUST PC-2
0800          ADJ3   EQU $F956      ;ADJUST PC-3
0800          REGD   EQU $FAD7      ;DISPLAY USER REGS
0800          RGDS   EQU $FADA      ;DISP REGS-NO CR
0800          GETL   EQU $FD67      ;GET INPUT LINE
0800          BL1    EQU $FE00      ;BLANK ROUTINE
0800          A1PC   EQU $FE75      ;COPY A1 TO PC
0800          BELL   EQU $FF3A      ;RING THE BELL
0800          RSTR   EQU $FF3F      ;RESTORE USER REGS
0800          SAVE   EQU $FF4A      ;SAVE USER REGS
0800          GETN   EQU $FFA7      ;GET ITEM,NONHEX
0800          TSUB   EQU $FFBE      ;PUSH AND GOTO SUB
0800          TSB1   EQU $FFC5      ;HANDLE THE MODE
0800          ZMOD   EQU $FFC7      ;ZERO THE MODE
0800          CHRT   EQU $FFCC      ; CHARACTER TABLE
0800          ;
0300                 ORG $0300
0300                 OBJ $0800
0300          ;
0300 D8       STRT   CLD            ;SET HEX MODE
0301 203AFF          JSR BELL       ;RING THAT CHIME
0304 A92A     CONT   LDA '*         ;LOAD INVERSE *
0306 8533            STA PRMP       ; AND STORE IN PRMP
0308 2067FD          JSR GETL       ;READ A LINE
030B 20C7FF          JSR ZMOD       ;SET MODE & Y=0
030E 20A7FF   NXTI   JSR GETN       ;GET ITEM,NONHEX
0311 8434            STY YSAV       ;CHAR IN A-REG
0313 C9EC     TRYS   CMP #$EC       ;IS IT STEP?
0315 F00B            BEQ ENT2       ;IF=STEP,GO ENT2
0317 C9ED     TRYT   CMP #$ED       ;IS IT TRACE?
0319 D00F            BNE TRCR       ;IF<>TRACE,TRYCR
031B AD00C0          LDA KBRD       ;WAS KEY PRESSD?
031E 3024            BMI AGIN       ;KEY ON,-->AGIN
0320 C634            DEC YSAV       ;MAKES STEP RPT
0322 20C7FF   ENT2   JSR ZMOD       ;ENTRY FOR STEP
0325 204903          JSR STPZ       ;GO STEP OUT
0328 101A            BPL AGIN       ;RTN TO INP LINE
032A C9C6     TRCR   CMP #$C6       ;IS IT A CR?
032C D009            BNE MCMD       ;IF<>CR,TRY MCMD
032E 20C5FF          JSR TSB1
0331 2000FE          JSR BL1        ;HANDLE CR AS BLNK
0334 4C0403          JMP CONT       ;RETURN TO CONT
0337 A017     MCMD   LDY #$17       ;TRY MONITOR CMDS
0339 88       CHRS   DEY            ;SEARCH MON CHARS
033A 30C4            BMI STRT       ;NOT FOUND, GO START
033C D9CCFF          CMP CHRT,Y     ;CMP WITH TABLE
033F D0F8            BNE CHRS       ;NOT FOUND, ->CHRS
0341 20BEFF          JSR TSUB       ;FND, CALL SUB
0344 A434     AGIN   LDY YSAV       ;RESTORE Y
0346 4C0E03          JMP NXTI       ;GET NEXT COMMAND
0349 2075FF   STPZ   JSR A1PC       ;ADR TO PC
034C 20D0F8   STEP   JSR DISA       ;TAKE ONE STEP
034F 68              PLA            ;ADJUST TO USER
0350 852C            STA RTNL       ;STACK AND SAVE
0352 68              PLA            ;RTN ADR
0353 852D            STA RTNH
0355 A208            LDX #$08
```

```
0357 BD   03     XQIN   LDA INM1,X        ;INIT XEQ AREA
035A 95                 STA XQT,X
035C CA                 DEX
035D D0F8               BNE XQIN
035F A13A               LDA (PCL,X)       ;JSR OPCODE BYTE
0361 F02C               BEQ XBRK          ;SPECIAL IF BREAK
0363 A42F               LDY LGTH          ;LENGTH FROM DASSY
0365 C920               CMP #$20
0367 F043               BEQ XJSR          ;HANDLE JSR,RTS,
0369 C960               CMP #$60          ;JMP,JMP( ),
036B F02F               BEQ XRTS          ; & RTI SPECIAL
036D C94C               CMP #$4C
036F F046               BEQ XJMP
0371 C96C               CMP #$6C
0373 F043               BEQ XJAT
0375 C940               CMP #$40
0377 F01F               BEQ XRTI
0379 291F               AND #$1F
037B 4914               EOR #$14
037D C904               CMP #$04          ;COPY USR INSTR
037F F002               BEQ XQ2           ;TO XEQ AREA
0381 B13A        XQ1    LDA (PCL),Y
0383 993C00      XQ2    STA XQT,Y
0386 88                 DEY
0387 10F8               BPL XQ1
0389 203FFF             JSR RSTR          ;RESTOR USR REGS
038C 4C3C00             JMP XQT           ;XEQ USER OP
038F 2082F8      XBRK   JSR INSD          ;PRINT USER PC
0392 20DAFA             JSR RGDS          ;AND REGS
0395 4C0003             JMP STRT          ;THEN GO STRT
0398 18          XRTI   CLC
0399 68                 PLA               ;SIMULATE RTI
039A 8548               STA STAT
039C 68          XRTS   PLA               ;RTS SIMULATION
039D 853A               STA PCL
039F 68                 PLA
03A0 853B        PCN2   STA PCH
03A2 A52F        PCN3   LDA LGTH          ;UPDAT PC BY LEN
03A4 2056F9             JSR ADJ3
03A7 843B               STY PCH
03A9 18                 CLC
03AA 9014               BCC NEWP
03AC 18          XJSR   CLC
03AD 2054F9             JSR ADJ2          ;UPDATE PC AND
03B0 AA                 TAX               ;PUSH ONTO STAK
03B1 98                 TYA               ;FOR JSR
03B2 48                 PHA               ;SIMULATION
03B3 8A                 TXA
03B4 48                 PHA
03B5 A002               LDY #$02
03B7 18          XJMP   CLC
03B8 B13A        XJAT   LDA (PCL),Y
03BA AA                 TAX               ;LOAD PC FOR JMP
03BB 88                 DEY               ;& (JMP)
03BC B13A               LDA (PCL),Y       ;SIMULATION
03BE 863B               STX PCH
03C0 853A        NEWP   STA PCL
03C2 B0F3               BCS XJMP
03C4 A52D        RTNJ   LDA RTNH
03C6 48                 PHA
```

```
03C7 A52C                LDA RTNL
03C9 48                  PHA
03CA 4CD7FA              JMP REGD         ;DISPLAY USR REG
03CD 18           BRAN   CLC              ;BRANCH TAKEN,
03CE A001                LDY #$01         ;ADD LEN+2 TO PC
03D0 B13A                LDA (PCL),Y
03D2 2056F9              JSR ADJ3
03D5 853A                STA PCL
03D7 98                  TYA
03D8 38                  SEC
03D9 B0C5                BCS PCN2
03DB 204AFF       NBRN   JSR SAVE         ;NORML RTRN AFTR
03DE 38                  SEC              ;EXQING USER OP
03DF B0C1                BCS PCN3         ;GO UPDATE PC
03E1 EA           INM1   NOP
03E2 EA           INIT   NOP
03E3 EA                  NOP              ;DUMMY FILL FOR
03E4 4CDB03              JMP NBRN         ;XEQ AREA
03E7 4CCD03              JMP BRAN
                        END

***** END OF ASSEMBLY


          ***************************
          *                         *
          * SYMBOL TABLE -- V 1.5   *
          *                         *
          ***************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

RTNL   002C  RTNH   002D  LGTH   002F  PRMP   0033  YSAV   0034  PCL    003A
PCH    003B  XQT    003C  STAT   0048

** ABSOLUTE VARABLES/LABELS

KBRD   C000  INSD   F882  DISA   F8D0
ADJ2   F954  ADJ3   F956  REGD   FAD7  RGDS   FADA  GETL   FD67  BL1    FE00
A1PC   FE75  BELL   FF3A  RSTR   FF3F  SAVE   FF4A  GETN   FFA7  TSUB   FFBE
TSB1   FFC5  ZMOD   FFC7  CHRT   FFCC  STRT   C300  CONT   0304  NXTI   030E
TRYS   0313  TRYT   0317  ENT2   0322  TRCR   032A  MCMD   0337  CHRS   0339
AGIN   0344  STPZ   0349  STEP   034C  XQIN   0357  XQ1    0381  XQ2    0383
XBRK   038F  XRTI   0398  XRTS   039C  PCN2   03A0  PCN3   03A2  XJSR   03AC
XJMP   03B7  XJAT   03B8  NEWP   03C0  RTNJ   03C4  BRAN   03CD  NBRN   03DB
INM1   03E1  INIT   03E2

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:01D2
```

TRACER: APPLE II 的一個除錯工具

作者：R. Kovacs

APPLE的STEP/TRACE 常規是非常方便的，但是如果您將它和此 TRACER 程式連起來使用，將會發現更具威力。

在APPLE II ROM 中的監督系統中含有許多有用的常規。包括記憶體的詰問與修改、鍵盤輸入、CRT 顯示輸出及卡帶 I/O 。除此之外，APPLE 並且還爲組合語言程式提供一些常規。一次處理的組合語言編譯程式（one-pass assembler）以及解組合語言程式（disassemble）在寫或看機器碼時非常有用。STEP/TRACE 的功能允許您控制程式的執行。

STEP 常規一次執行一個指令並顯示它的位址，包括所翻譯的碼，暫存器A、X、Y、P的值，以及堆疊指標。您可以修改任何暫存器的值，並且可以從下一個或任意一個指令繼續執行。

很不幸的，將這些值通通打在螢幕上只能顯示最近的11個指令的狀況。但我們若犧牲一些其他資料而多印一些 PC

的經歷值，將會更有用。

程式部份

此 TRACER 程式在使用時要配合 APPLE的STEP/TRACE 常規，以加強它們的功能。基本上這程式的作用主要在控制螢幕的輸出。此常規將印出最近執行的160個指令的 PC 值。只有最後一個指令的所有資料都會印出（即包括暫存器的值等）。您可以繼續執行或暫時停止來修改暫存器或記憶體的值。TRACER 也有可能碰到中斷碼（ 00 ），碰到時會發出二聲嗶嗶，然後就等使用者的動作。中斷點之前所執行的最後一個指令也會印出。

注意： TRACER 延遲一個指令顯示。如果經由 RESET 進入監督系統，而由於下一個指令已經執行，所以要保存現今暫存器的值很困難。您可以利用監督系統命令 Control−E 查看值對不對。

加有說明的程式報表在後面。程式從 $ 300 開始，計有 190 個位元組。它沒有用到第 0 頁記憶體的其他位置。

它如何工作

TRACER將STEP/TRACE 所產生的字元串處理一

下，然後控制那些資訊顯示在螢光幕上。TRACER 將尋找某一特定字元和順序，以便知道何時才是一個指令的結束。

APPLE所用的 2 行顯示格式增加了一點複雜性。執行完一個指令之後，以一Return（ $8D）爲開頭，先印出十六進位的位址，此位址即剛剛被執行的指令的所在位址，再印出其他的字元串。當此行印完後，又接著一個 Return ，然後再印第二行。

TRACER 找到 Return 碼之後，經由 COUT 傳送口將第一行的資料轉移到 TRACER 。剩餘的資料則存在緩衝器（ Buffer ）中。而第二行的資料是一個 Return 碼緊接着一個空白碼（ $A0 ）來區別。等到下一個 Return 碼出現時，便利用監督系統的 COUT 常規（ $FDF0 ）將緩衝器中的十六進位的位址（加一個空白）顯示在螢幕上。此常規將會以每劃 8 個位址一共 20 列的格式，顯示160 個位址。

因爲緩衝器碰巧是螢幕記憶體的一部份，所以它也會顯示出來。經由移動自動捲頁窗口（ scrolling window ）的底界來保護緩衝器部份。

Control Y 的功能是利用在位址 $3F8 的 JMP 指令啓動 TRACER 。它將清除螢幕，設立自動捲頁窗口，並且設立在 $36 與 37 的COUT傳送口，以便將資料傳到 TRACER 。

使用法

TRACER 的用法很簡單：

1. 從$300位址開始載入TRACER。（不要忘記Control-Y在3FB上的JMP：4C 00 03）。
2. 從監督系統執行程式只要打：Y。XXXXT，其中Y。就是Control-Y，XXXX 就是開始除錯的位址。螢幕上將會很乾淨，TRACER已經由COUT連結好了，並且從指令位址開始追踪。
3. 開始時，TRACER 只追踪一個指令，並且在螢幕的底端顯示 STEP/TRACE 的資料後會暫停。若要繼續追踪只要按空白鍵即可。剛剛被執行的指令位址也會出現在螢幕最上面。
4. 若按 Return 鍵，則將會執行一整頁的指令，按下任何鍵將再取得控制權。
5. 當然，按下Reset鍵將會回到監督系統，但仍可處理暫存器和記憶體的值。再打 Y。T 便可再進入 TRACER。

較早的

↑

160個已執行過的指令位址

↓

最近的

```
FCA9 FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAE
FCAF FCB1 FCA9 FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAE
FCAF FCB1 FCA9 FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAE FCAF FCB1
FCA9 FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAE FCAF FCB1 FCA9 FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAC FCAA FCAC FCAE FCAF FCB1 FCA9 FCAA
FCAC FCAA FCAC FCAA FCAC FCAA FCAC FCAA
FCAA-    E9 01         SBC    #$01
 A=05 X=00 Y=00 P=31 S=99
```

正常的STEP/TRACE輸出

圖 1　這個例題說明TRACER處理APPLE WAIT常規（$FCA8）的輸出結果。正常的STEP/TRACE輸出結果在螢幕的最下面，其上是160個指令的位址。

```
0800    ;**************************
0800    ;*                        *
0800    ;*        TRACER          *
0800    ;*       R. KOVACS        *
0800    ;*                        *
0800    ;*  COPYRIGHT (C) 1981    *
0800    ;*    MICRO INK, INC.     *
0800    ;* CHELMSFORD, MA 01824   *
0800    ;*  ALL RIGHTS RESERVED   *
0800    ;*                        *
0800    ;**************************
0800    ;
0800    ;
```

```
0800                 ;ENTER VIA CONTROL-Y FOLLOWED BY XXXXT
0800                 ;WHERE XXXX IS THE ADDRESS TO BEGIN TRACING
0800                 ;
0800                 WNDBTM EPZ $23            ;BOTTOM OF SCROLLING WINDOW
0800                 PCL    EPZ $3A            ;PGM COUNTER
0800                 ;
0800                 WINDOW EQU $FB3C          ;SET NORMAL SCROLL WINDOW
0800                 BELL   EQU $FBDD          ;TOGGLE SPEAKER
0800                 CLEAR  EQU $FC58          ;CLEAR SCREEN, HOME CURSOR
0800                 COUT   EQU $FDFC          ;OUTPUT CHAR TO SCREEN
0800                 READ   EQU $C000          ;KEYBOARD STROBE
0800                 RESET  EQU $C010          ;RESET KEYBOARD
0800                 ;
0800                 BUFF   EQU $0750          ;LINE #22-COL #0
0800                 BUFF1  EQU $07D0          ;     #23     #0
0800                 ;
0800                 ;************************************
0800                 ;SET UP CONTROL-Y JUMP TO $3F8
0800                 ;
03F8                        ORG $03F8
03F8                        OBJ $08F8
03F8                 ;
03F8 4C0003                 JMP TRINIT
03FB                 ;
03FB                 ;************************************
03FB                 ;
03FB                 ;TRACER INITIALIZATION
03FB                 ;
0300                        ORG $0300
0300                        OBJ $0800
0300                 ;
0300 203CFB          TRINIT JSR WINDOW         ;CLEAR ENTIRE SCREEN
0303 2058FC                 JSR CLEAR
0306 A915                   LDA #$15           ;SET SCROLL WINDOW
0308 8523                   STA WNDBTM
030A A91C                   LDA #TRACER        ;SET COUT HOOK
030C 8536                   STA $36            ;TO TRACER
030E A903                   LDA /TRACER
0310 8537                   STA $37
0312 A91F                   LDA #$1F           ;INIT CH FOR EVEN PACING
0314 8524                   STA $24
0316 A902                   LDA #$02           ;INIT PGCNT FOR
0318 8DBB03                 STA PGCNT          ;SINGLE STEP
031B 60                     RTS
031C                 ;
031C                 ;************************************
031C                 ;
031C 8DB703          TRACER STA SAVEA          ;SAVE A & Y
031F 8CB803                 STY SAVEY          ;REGISTERS
0322 2CBA03                 BIT CRFLG          ;WAS LAST CHAR A CR?
0325 301C                   BMI CR             ;YES
0327 C98D                   CMP #$8D           ;IS THIS CHAR A CR?
0329 F00C                   BEQ SETCR          ;YES
032B ACB903          STORE  LDY BPTR           ;LOAD BUFF POINTER
032E 995007                 STA BUFF,Y         ;NO, SO STORE IT
0331 C8                     INY                ;INC POINTER
```

```
0332 8CB903           STY BPTR         ;  & SAVE IT
0335 D005             BNE DONE         ;BRANCH ALWAYS
0337 A080    SETCR    LDY #$80         ;SET CR FLAG
0339 8CBA03           STY CRFLG
033C ADB703  DONE     LDA SAVEA        ;RESTORE
033F ACB803           LDY SAVEY        ;REGISTERS
0342 60               RTS              ;RETURN TO MONITOR
0343 A000    CR       LDY #$00         ;RESET CR FLAG
0345 8CBA03           STY CRFLG
0348 C9A0             CMP #$A0         ;IS NEXT CHAR A SPACE?
034A D007             BNE ADDR-2       ;NO
034C A080             LDY #$80         ;ADJ PTR TO NEXT
034E 8CB903           STY BPTR         ;LINE ON SCREEN
0351 D0D8             BNE STORE        ;BRANCH ALWAYS
0353 A000             LDY #$00         ;INIT BUFF POINTER
0355 B95007  ADDR     LDA BUFF,Y
0358 20F0FD           JSR COUT         ;OUTPUT IT
035B C8               INY
035C C004             CPY #$04         ;FINISHED PRINTING 4 CHAR
035E 90F5             BCC ADDR         ;NO
0360 A9A0             LDA #$A0
0362 20F0FD           JSR COUT         ;OUTPUT A SPACE
0365                  ;
0365                  ;CHECK FOR BREAK
0365                  ;
0365 A000             LDY #$00
0367 B13A             LDA (PCL),Y      ;GET OPCODE
0369 F00C             BEQ KEY1         ;PAUSE IF BREAK
036B                  ;
036B                  ;LOOK FOR KEYBOARD INPUT
036B                  ;
036B CEBB03  KEY      DEC PGCNT        ;CHECK PAGING
036E F00D             BEQ KEY2
0370 2C00C0           BIT READ         ;ANY KEYBOARD INPUTS?
0373 300D             BMI KEY3         ;YES
0375 1020             BPL TRACE
0377 20DDFB  KEY1     JSR BELL         ;SOUND BELL FOR BRK
037A 20DDFB           JSR BELL
037D A0A0    KEY2     LDY #$A0         ;RESET PAGE COUNTER
037F 8CBB03           STY PGCNT        ;AND PAUSE
0382 8D10C0  KEY3     STA RESET
0385 2C00C0  KEY4     BIT READ         ;LOOP UNTIL ANOTHER
0388 10FB             BPL KEY4         ;KEY IS HIT
038A                  ;
038A                  ;TEST INPUT FOR TRACE, STEP OR QUIT
038A                  ;
038A AD00C0           LDA READ         ;LOAD CHARACTER
038D C98D             CMP #$8D         ;'RETURN' TO CONTINUE TRACE
038F F006             BEQ TRACE
0391 C9A0             CMP #$A0         ;'SPACE' TO SINGLE STEP
0393 F005             BEQ STEP
0395 D0E3             BNE KEY1+3       ;NO MATCH, TRY AGAIN
0397 8D10C0  TRACE    STA RESET        ;RESET KEYBOARD STROBE
039A EA      STEP     NOP
039B                  ;
039B                  ;FILL PROTECTED FIELD WITH SPACES
039B                  ;
039B A9A0             LDA #$A0         ;ASCII SPACE
```

```
039D A027     128           LDY #$27        ;40 CHAR/LINE
039F 995007   129    FILL   STA BUFF,Y
03A2 99D007   130           STA BUFF1,Y
03A5 88       131           DEY
03A6 10F7     132           BPL FILL
03A8          133    ;
03A8 ADB703   134           LDA SAVEA
03AB A000     135           LDY #$00        ;RESET BUFF POINTER
03AD 8CB903   136           STY BPTR
03B0 C9B0     137           CMP #$B0        ;IS 1ST CHAR 0-9/A-F ?
03B2 9088     138           BCC DONE        ;NO
03B4 4C2B03   139           JMP STORE       ;YES, OUTPUT IT
03B7          140    ;
03B7          141    ;
03B7 00       142    SAVEA  HEX 00
03B8 00       143    SAVEY  HEX 00
03B9 00       144    BPTR   HEX 00
03BA 00       145    CRFLG  HEX 00
03BB 00       146    PGCNT  HEX 00
             147           END

***** END OF ASSEMBLY


          ***************************
          *                         *
          * SYMBOL TABLE -- V 1.5   *
          *                         *
          ***************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

WNDBTM 0023  PCL    003A

** ABSOLUTE VARABLES/LABELS

WINDOW FB3C  BELL   FBDD  CLEAR  FC58 COUT   FDF0
READ   C00C  RESET  C010  BUFF   0750 BUFF1  07D0 TRINIT 030C TRACER 031C
STORE  032B  SETCR  0337  DONE   033C CR     0343 ADDR   0355 KEY    036B
KEY1   0377  KEY2   037D  KEY3   0382 KEY4   0385 TRACE  0397 STEP   039A
FILL   039F  SAVEA  03B7  SAVEY  03B8 BPTR   03B9 CRFLG  03BA PGCNT  03BB


SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0102
```

APPLE整數BASIC副常規PACK和LOAD

作者：Richard F. Suitor

許多程式設計師常常會寫一些機器語言副常規，而由整數 BASIC 程式叫用。而將這些副常規和 BASIC程式一起存在一個檔案（File）中，這可能會造成混亂。本程式將使您能在一個檔案中包含機器語言副常規和 BASIC 程式，並且可以很容易的存到磁碟或磁帶上。

在CONTACT的第一期中（APPLE的時事通訊），曾建議將組合語言常規和 BASIC 程式一起載入記憶體中。簡單的說，將組合語言部份底下的 BASIC 指標丟棄，而加入一個 BASIC 指令便能恢復此指標。此程序很簡單且有效率，但有二個限制。第一，如果 BASIC 程式和副常規分得很開將很不方便（且如果副常規剛好從位址 $ 800 開始，即記憶體的顯示部份，將會很難處理）。第二，用此方法存檔的程式不能使用其他HIMEM，因此很不方便。

在此所提的副常規曾多花一些功夫來避免這些問題。它是完全和位置無關的，它可以利用監督系統的移動（move）

命令搬移至主記憶體中任意位置，並且不會因位置改變而需要修改。它將擴大SWEET 16的使用範圍，SWEET 16是APPLE監督系統 ROM 所提供的16- 位元翻譯程式。

如何使用PACK和 LOAD

從APPLE整數BASIC來利用CALL MKUP 使用此常規，其中 MKUP 是128（十進位）加上此常規的開頭位址。提示符號爲“ @ ”。在此時打入程式儲存的界限，如BBBB.EEEE（BBBB是起始位址，EEEE是最後位址；和監督系統用法一樣）。一行可以打入好幾組，只要用空白分開，或打成不同行。當最後一組做完後，打入S ，便可回到 BASIC 。程式現在便可存起來了。

若要載入記憶體，則進入 BASIC ，使用 LOAD 命令。做完之後，打入 RUN 。這個 RUN 命令，將把所有的常規移回原來的位置，並回到 BASIC ，而不會眞正執行程式；下一個 RUN 命令就會執行了。

若是在召用 MKUP之後，第一次ROM之前，將程式列印出來，將只會看到一個 BASIC 指令（此指令是用來做回復動作的），並且會有亂叫聲。假如做了上述動作，則打CTRL－C ，再RESET ，就會回到BASIC 。

注意1：常規必須存在記憶體中較安全的位置，使得在做群集（PACK）動作時，不會有重寫現象（ overwriting ）

發生。常規的起始位置必須在HIMEM之上（即在高解析度顯示區）或是在 BASIC 程式起始位置的底下 $17A+4*N+W 個位元組後的位置，其中N是常規的個數，W是所有常規所佔的位元組。並且存在記憶體較高位置的常規要先做群集動作，才不會發生重寫現象。

注意 2：在召用 MKUP 之後，不要去編輯修改程式。如果一定要，則先 RUN 一次將程式解集（ unpack ），再修改，然後再召用 MKUP 。

PACK和LOAD如何工作

此常規首先將 BASIC 程式底下的常規群集在一起。然後視需求再群集其他的常規，只要輸入起始位置和所佔的位元組長度。當打入 S 之後，它將集合一些資料恢復LOMEM，並回到 BASIC 程式。常規的前 $46 個字將形成一個 BASIC 敍述，當打 RUN 時會處理回復動作。

如果程式需要一個特別的HIMEN（例如，高解析度程式用的），必須在 LOAD 之前加入。在作回復動作時，將會把召用 MKUP 時所設的 LOMEM 的值重新設定。

一些便利的載入和進入點爲：

BASO（載入）十六進位	MKUP 十六進位	進入點 十進位
800	880	2176
A90	B10	2832
104C	10CC	4300
2050	20D0	8400
3054	30D4	12500
6000	6080	24704
9000	9080	−28544

在磁碟上，BLOADS 在 $9400，MKUP 在 $9480，十進位是 −27520。

副常規PACK和LOAD的附錄

當寫好 PACK 和 LOAD 副常規時，我想我應該可以提供使用者在執行程式之前，能很容易的群集和解集副程式（那時只有卡帶可以用）。用了一陣子後，大家都認爲當一個程式已經沒有錯時，如果能在一個動作內做完解集（unpack）和執行的動作將會更好。Alan Hill 告訴我說，如果我用 JMP 到 $EFEC 而不是到 $E003，便可完成此工作。

很不幸的，在我寫此程式時，並沒有想到此改進方法。此常規回到 ROM 的位址應該是不會變的，如果 APPLE

PLUS 的所有者拿到整數 BASIC 的群集程式將不能使用。

密封（ enclosed ）的常規將會解決它們的問題，以及想要自動RUN所需要改變的回應向量（ return vector ）。這常規將位址改變成 UNPACK 程序完成時所送回的位址。

所需要的位址放在位置 0 和 1 上。例如，假如您想用位址 $EFEC，從監督系統打入：

```
* O : EC EF
```

或從BASIC打入：

```
POKE 0,236
POKE 1,239
```

爲了完成此改變動作，此常規和所要改變的程式都要在記憶體中。程式必須 LOAD ，但不執行。此常規在位址 800（ $320 ），但可以在任何位置執行。BLOAD 此常規，設立位置 0 和 1 的值，再 CALL 800 完成改變動作。您便可以把已改變過的程式存起來。

您可能用到的位址爲：

目　　的	ROM 形式	DISK 形式
回到 BASIC	$E003	$03D0
UNPACK和 RUN	$EFEC	（$9D58）

最後一個進入點（從磁碟上做 UNPACK 和 RUN 動作）表示您將＄9D58 的內容存入 0，＄9D59 的內容存入 1。這個方法在 APPLE 版的整數 BASIC 上使用。請注意，雖然對任何 48K 以磁碟爲主的系統，＄9D58、＄9D59 均相同，但內容可能不同。所以有時程式不能在別的系統上執行。如果使用位址＄3D0 便能被普遍的使用。但這個選擇有缺點，即它不會同時做 UNPACK 和 RUN 動作，但如果在卡帶系統上或第 3 頁已重寫的磁碟系統都不能使用。對這些系統，進入監督系統，再打入 3D0：4C03E0 。（注意：這將啓動 3D0G 回到 BASIC ，但不會回復一個不連接的 DOS ）

然而，使用此程式常規，任何“群集”的程式都能載入並且在使用者的系統上執行，以及存檔。

注意：此 PACK－LOAD 常規在執行時，SWEET－16 必須在您的 APPLE 中。縱使改成附錄所提的型式，如果 SWEET－16 不存在，則將不能執行。

```
0800      ;*************************
0800      ;*                       *
0800      ;* PACK AND LOAD SUBRTN  *
0800      ;*  RICHARD F. SUITOR    *
0800      ;*                       *
0800      ;*       PACK-LOAD       *
0800      ;*                       *
0800      ;*  COPYRIGHT (C) 1981   *
0800      ;*    MICRO INK, INC.    *
0800      ;* CHELMSFORD, MA 01824  *
0800      ;*  ALL RIGHTS RESERVED  *
0800      ;*                       *
0800      ;*************************
0800      ;
0800      ;
0800      ; INTEGER BASIC ROUTINE TO PACK AND RELOAD
0800      ; MACHINE LANGUAGE SUBROUTINES AND/OR TABLES
```

```
0800                ;
0800                ; CALL BASO+128(DEC) = MKUP TO PACK EXISTING
0800                ; ROUTINES AT THE START OF BASIC
0800                ;
0800                ; RUNNING THE PACKED PROGRAM WILL UNPACK THE
0800                ; PACKED ROUTINES AND RETURN TO BASIC (>)
0800                ;
0800                ; CHANGE THE LAST INSTRUCTION OF THE LISTING
0800                ; TO 'JMP BRUN' TO UNPACK AND RUN IN ONE OPERATION.
0800                ; NOTE: THIS STEP NOT NORMALLY TAKEN UNTIL
0800                ; PROGRAM DEVELOPMENT IS COMPLETE!
0800                ;
0800                ; PROGRAM WILL RUN ANYWHERE IN MEMORY
0800                ;
0800                ;
9400                       ORG $9400
9400                       OBJ $800
9400                ;
9400                ;
9400                ACCL   EPZ $00             ;R0, ACCUMULATOR
9400                BSCL   EPZ $02             ;R1
9400                TABL   EPZ $04             ;R2
9400                TBCL   EPZ $06             ;R3
9400                HIMS   EPZ $08             ;R4
9400                LMRT   EPZ $0A             ;R5
9400                BPRG   EPZ $0C             ;R6
9400                FRML   EPZ $0E             ;R7
9400                NBYT   EPZ $10             ;R8
9400                BPR2   EPZ $12             ;R9
9400                PTLL   EPZ $14             ;R10
9400                XTAB   EPZ $16             ;R11
9400                SKPL   EPZ $18             ;R12, SW16 STACK PTR
9400                MODE   EPZ $31
9400                YSAV   EPZ $34
9400                PRMP   EPZ $33             ;PROMPT
9400                LMML   EPZ $4A             ;INTEGER LOMEM
9400                HIML   EPZ $4C             ;INTEGER HIMEM
9400                LMWL   EPZ $CC             ;INTEGER END OF VARIABLES
9400                BBSL   EPZ $CA             ;BOTTOM OF PROGRAM
9400                JSRL   EPZ $CE             ;CALL VECTOR
9400                BSC2   EQU $E003           ;BASIC
9400                BRUN   EQU $EFEC           ;RUN BASIC
9400                BUFF   EQU $0200           ;INPUT BUFFER
9400                SW16   EQU $F689           ;SWEET16 ENTRY
9400                GTNM   EQU $FFA7           ;GET # FROM BUFF.
9400                PBL2   EQU $F94A           ;PRINT BLANKS
9400                COUT   EQU $FDED           ;OUTPUT CHAR.
9400                BELL   EQU $FF3A           ;BEEP
9400                GTLN   EQU $FD67           ;INPUT A LINE
9400                ;
9400                ; BASIC STATEMENT TO START CODE REPLACEMENT
9400                ; PROCESS...
9400                ;
9400                ; 0 POKE 1,76: POKE 2,(PEEK(202)+70) MOD 256:
9400                ;   POKE 3,(PEEK(203) + (PEEK(202)+70)/256):
9400                ;   CALL 1
9400                ;
9400 4600C0         BASO   HEX 4600C0
9403 64B1C1                HEX 64B1C1
9406 0065B7                HEX 0065B7
9409 4C0003                HEX 4C0003
940C 64B2                  HEX 64B2
```

```
940E 020065          HEX 020065
9411 382E3F          HEX 382E3F
9414 B2CA            HEX B2CA
9416 007212          HEX 007212
9419 B74600          HEX B74600
941C 721F            HEX 721F
941E B20001          HEX B20001
9421 0364B3          HEX 0364B3
9424 0300            HEX 0300
9426 65382E          HEX 65382E
9429 3FB2CB          HEX 3FB2CB
942C 0072            HEX 0072
942E 12382E          HEX 12382F
9431 3FB2CA          HEX 3FB2CA
9434 0072            HEX 0072
9436 12B746          HEX 12B746
9439 007215          HEX 007215
943C B200            HEX B200
943E 017203          HEX 017203
9441 4DB101          HEX 4DB101
9444 0001            HEX 0001
9446            ;
9446            ; INITIALIZE POINTERS
9446            ;
9446 D8         PTBK CLD
9447 A201            LDX #1
9449 B5CA       PT02 LDA BBSL,X         ;R1 IS START OF PACKED PROG.
944B 9502            STA BSOL,X
944D B54C            LDA HIML,X         ;R4 IS END (HIMEM)
944F 9508            STA HIMS,X
9451 CA              DEX
9452 10F5            BPL PT02
9454 2089F6          JSR SW16
9457 105201          SET R0,PTLP-BASO
945A 185701          SET R8,PTLP+5-BASO
945D A1              ADD R1             ;SET R7 TO CURRENT START OF
945E 37              STO R7             ; PACKED DATA (BBSL+PTLP-BASO)
945F 67              LDD @R7
9460 35              STO R5             ; PUT IN R5
9461 67              LDD @R7            ;PUT ORIGINAL LENGTH OF PROGRAM
9462 36              STO R6             ; IN R6
9463 24              LDR R4
9464 B6              SUB R6             ;CALCULATE START OF ORIGINAL
9465 36              STO R6             ; PROGRAM AND PUT IN R6
9466 1A1100          SET RA,ST16+1-PLP1
9469 BA              SUB RA             ;CURRENT LOCATION OF ENTRY
946A 3A              STO RA             ;TO RESTORE LOOP IN RA
946B 67              LDD @R7            ;BASO LOCATION TO LEAVE ROUTINE
946C 33              STO R3
946D 00              RTN
946E A201            LDX #1
9470            ;
9470            ; RESTORE ORIGINAL LOMEM AND START
9470            ; OF ORIGINAL PROGRAM...
9470            ;
9470 B50A       PT04 LDA LMRT,X
9472 954A            STA LMML,X
9474 95CC            STA LMWL,X
9476 B50C            LDA BPRG,X
9478 95CA            STA BBSL,X
947A CA              DEX
947B 10F3            BPL PT04
947D 6C1400          JMP (PTLL)
```

```
9480                ; JMP (RA) = PLP1
9480                ;
9480                ; SECTION TO PERFORM PACK
9480                ;
9480 A201           MKUP   LDX #1
9482 B54A           MK21   LDA LMML,X
9484 950A                  STA LMRT,X          ;R5=LOMEM
9486 B5CA                  LDA BBSL,X          ;R9,R6=START
9488 9512                  STA BPR2,X          ; OF PROGRAM
948A 950C                  STA BPRG,X
948C B5CE                  LDA JSRL,X          ;R2=MKUP LOCATION
948E 9504                  STA TABL,X
9490 B54C                  LDA HIML,X          ;R4=HIMEM
9492 9508                  STA HIMS,X
9494 CA                    DEX
9495 10EB                  BPL MK21
9497                ;
9497                ; INIT AND PACK THE RESTORE LOOP AT PTLP
9497                ;
9497 2089F6                JSR SW16
949A 24                    LDR R4
949B B9                    SUB R9
949C 39                    STO R9              ;LENGTH OF PROGRAM
949D 118000                SET R1,MKUP-BASO
94A0 22                    LDR R2
94A1 B1                    SUB R1
94A2 31                    STO R1              ;BASO LOCATION
94A3 105201                SET R0,PTLP-BASO
94A6 A1                    ADD R1
94A7 32                    STO R2              ;PTLP LOCATION
94A8 181800                SET R8,ST16-PTLP
94AB A8                    ADD R8
94AC 33                    STO R3              ;ST16 LOCATION
94AD E3                    INR R3              ;END (ST16) + 1
94AE 1C5000                SET RC,$50          ;SW16 STACK
94B1 0C42                  BSB MV52            ;PACK RESTORE LOOP
94B3 00             MK22   RTN
94B4 A9C0           MK01   LDA #$C0            ;'@'
94B6                ;
94B6                ; GET LIMITS AND PACK PROGRAMS
94B6                ;
94B6 8533                  STA PRMP            ;PROMPT IS '@'
94B8 A900                  LDA #0
94BA 8521                  STA MODE
94BC 2067FD                JSR GTLN            ;GET COMMAND
94BF 8616                  STX XTAB            ;END OF COMMAND
94C1 A000                  LDY #0
94C3 B90002                LDA BUFF,Y
94C6 C9D3                  CMP #$D3            ;'S', STOP?
94C8 F068                  BEQ MK10            ;YES
94CA 20A7FF         MK06   JSR GTNM            ;START OF RANGE
94CD C9A7                  CMP #$A7            ;F(.) (SEE MON.)
94CF F010                  BEQ MK02
94D1 98             MERR   TYA                 ;ERROR IF HERE
94D2 AA                    TAX
94D3 204AF9                JSR PBL2            ;ERROR INDICATOR
94D6 A9DE                  LDA #$DE            ;'^'
94D8 20EDFD                JSR COUT
94DB 203AFF                JSR BELL
94DE 18             MK05   CLC
94DF 90D3                  BCC MK01
94E1 E631           MK02   INC MODE
94E3 20A7FF                JSR GTNM            ;END OF RANGE
```

```
94E6                ;
94E6                ; A1 & A3 NOW HAVE 1ST #, A2 SECOND
94E6                ; SET UP MOVE TO JUST BELOW (BBSL)
94E6                ; AND LOWER BBSL
94E6                ;
94E6 2089F6                 JSR SW16
94E9 011E                   BRA SM02
94EB 183C00         MV51    SET R8,$3C
94EE 68                     LDD @R8
94EF 32                     STO R2          ;R2=A1
94F0 68                     LDD @R8
94F1 33                     STO R3          ;R3=A2
94F2 B2                     SUB R2          ;A2-A1
94F3 38                     STO R8
94F4 E3                     INR R3
94F5 83             MV52    POP @R3         ;MOVE FROM (R3) DOWN TO (R2)
94F6 96                     STP @R6         ;TO (R6) AND DOWN
94F7 23                     LDR R3
94F8 D2                     CPR R2
94F9 07FA                   BNZ MV52
94FB 28                     LDR R8          ;LENGTH-1
94FC 33                     STO R3
94FD 180800                 SET R8,8
9500 88                     POP @R8         ;PREFACE PACKED ROUTINE
9501 96                     STP @R6         ;BY LENGTH-1 AND BY
9502 88                     POP @R8         ;STARTING ADDRESS
9503 96                     STP @R6
9504 88                     POP @R8
9505 96                     STP @R6
9506 88                     POP @R8
9507 96                     STP @R6
9508 0B                     RSB
9509 0CE0           SM02    BSB MV51
950B 00             SM03    RTN
950C C9EC           MK09    CMP #$EC        ;F(S) STOP?
950E F022                   BEQ MK10        ;YES
9510 C9C6                   CMP #$C6        ;F(CR) END OF LINE?
9512 F0A0                   BEQ MK01        ;YES, GET NEW COMM.
9514 C999                   CMP #$99        ;F( ) ;BLANK?
9516 F003                   BEQ MK12        ;YES
9518 D0B7                   BNE MERR        ;ERROR IF OTHER
951A C8             MK11    INY
951B B90002         MK12    LDA BUFF,Y      ;GET NEXT COMM. CHAR
951E C416                   CPY XTAB        ;END OF LINE?
9520 B092                   BCS MK01        ;YES, GET ANOTHER
9522 C9AC                   CMP #$A0        ;BLANK
9524 F0F4                   BEQ MK11
9526 C98D                   CMP #$8D        ;CR.
9528 F08A                   BEQ MK01
952A C9D3                   CMP #$D3        ;'S'
952C F004                   BEQ MK10
952E C631                   DEC MODE
9530 F098                   BEQ MK06        ;ALWAYS
9532                ;
9532                ; PACK 1ST PART AND CLEAN UP
9532                ;
9532 2089F6         MK10    JSR SW16
9535 21                     LDR R1
9536 32                     STO R2          ;BASO LOCATION
9537 185201                 SET R8,PTLP-BASO
953A A8                     ADD R8
953B 37                     STO R7          ;PTLP LOCATION
953C 25                     LDR R5          ;PACK:
```

```
953D 77              STD @R7              ; LOMEM
953E 29              LDR R9
953F 77              STD @R7              ; ORIGINAL LENGTH OF PROGRAM
9540 21              LDR R1               ; BASC LOCATION
9541 77              STD @R7              ;ONTO END OF 1ST PART
9542 27              LDR R7               ;OF ROUTINE
9543 33              STO R3
9544 0CAF            BSB MV52             ;PACK BASC-PTLP PLUS ABOVE VARS.
9546 66        SM04  LDD @R6              ;STRIP PREFACE
9547 66              LDD @R6              ;LEAVING BASIC STATEMENT
9548 00              RTN
9549 A50C            LDA BPRG
954B 85CA            STA BBSL             ;R6 IS NEW START
954D A50D            LDA BPRG+01          ;OF PROGRAM
954F 85CB            STA BBSL+01
9551 60              RTS
9552           ;
9552           ; RESTORE LOOP --THIS LOOP DOES THE ACTUAL
9552           ; UNPACKING AND IS ALWAYS JUST IN FRONT OF
9552           ; THE ORIGINAL BASIC PROGRAM...
9552           ;
9552 2089F6    PTLP  JSR SW16
9555 61        PLP0  LDD @R1
9556 33              STO R3               ;DESTINATION
9557 61              LDD @R1
9558 38              STO R8               ;LENGTH
9559 00              RTN
955A 2089F6    PLP1  JSR SW16
955D 41        MV60  LDR @R1              ;UNPACK
955E 53              STO @R3
955F F8              DCR R8
9560 04FB            BIP MV60
9562 21              LDR R1
9563 D6              CPR R6               ;AT END YET?
9564 05EF            BIM PLP0             ;NOT YET
9566 00        PLP2  RTN
9567 4C03E0          JMP BSC2
956A           ; OR JMP BRUN TO RUN AUTOMATICALLY
956A 00        ST16  HEX 00
                     END

***** END OF ASSEMBLY


          ****************************
          *                          *
          * SYMBOL TABLE -- V 1.5    *
          *                          *
          ****************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

ACCL    000C  BSOL    0002  TABL    0004  TBCL    0006  HIMS    0008  LMRT    00CA
BPRG    000C  FRML    000E  NBYT    0010  BPR2    0012  PTLL    0014  XTAB    0016
SKPL    0018  MODE    0031  YSAV    0034  PRMP    0033  LMML    004A  HIML    004C
LMWL    00CC  BBSL    00CA  JSRL    00CE

** ABSOLUTE VARABLES/LABELS
```

```
BSC2   E003  BRUN  EFEC  BUFF  0200
SW16   F689  GTNM  FFA7  PBL2  F94A  COUT  FDED  BELL  FF3A  CTLN  FD67
BASO   9400  PTBK  9446  PT02  9449  PT04  9470  MKUP  9480  MK21  9482
MK22   94B3  MK01  94B4  MK06  94CA  MERR  94D1  MK05  94DE  MK02  94E1
MV51   94EB  MV52  94F5  SM02  9509  SM03  950B  MK09  950C  MK11  951A
MK12   951B  MK10  9532  SM04  9546  PTLP  9552  PLPO  9555  PLP1  955A
MV60   955D  PLP2  9566  ST16  956A

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:01DA
```

MEAN-14: APPLE II 的虛假浮點處理機

作者：R. M. Mottola

繼 SWEET 16 之後，此程式將提供處理 5 個位元組的浮點數值算術動作。這個“處理機”將可簡化並加強您的算術運算能力。

在APPLE II 電腦發展的初期，都利用韌體（Firmware）來克服一些障礙。就我們所知，6502 是一個 8 位元的微處理機，但常常有一些常規需要二倍精準度整數的運算動作。我們可利用重複動作來完成此目的，但是效率比較差。因此若要運算二倍精準度的整數，我們所需要的是一個能夠處理二個位元組整數的算術運算處理機。所以，一個虛假的處理機就是一個機器語言程式，而它動作就像處理機一樣。

“SWEET 16 虛假的翻譯器”就是很好的答案，並且爲許多APPLE 使用者使用。它存在於APPLE II 的FO 整數 BASIC ROM 中，由 $F689 到F7FA 。從軟體的觀點來看，使用此翻譯器時就好像在使用微處理機一樣。要設計此程式必須用到不同的指令和運算元（operands）。若

用人直接翻譯也很容易，因爲指令集並不多，且運算符號（ operators ）的格式也很直接。Randall Hyde 所寫的 Lisa 組合語言翻譯器很受歡迎，它也能翻譯 SWEET 16 的助憶符號（ mnemonics ）。

MEAN 14 是在 SWEET 16 之後發展出來的。它設計得像一個硬體處理機。MEAN 14 可以處理許多 5 個位元組的浮點數值的運算，而不是用來處理二個位元組的整數。這些值都是以 APPLESOFT 變數表示。

在 APPLE II PLUS 或使用 APPLESOFT ROM 卡的標準 APPLE II ，MEAN 14 處理機都能處理浮點機器語言程式。因爲 APPLE 在 APPLESOFT 的浮點常規方面都沒有報告文件，所以若想在組合語言程式中寫一些浮點常規將會很困難。如果要執行複雜或重複函數時，縱使知道常規的位置和進入點也是只有一點用處而已。當然，您也可以將一些複雜的函數用 APPLESOFT 的 BASIC 寫，但是 MEAN 14 在執行時幾乎快 10 倍或者更多。理由很簡單，APPLESOFT 在翻譯時比 MEAN 14 麻煩。現在以二個數相加爲例子，如果使用 APPLESOFT ，則將此二數以變數表示，首先 APPLESOFT 必須按排它們的位置。如果是陣列資料，則必須往上移再找位置給新的變數。在機器方面，這些動作將花費很多時間。

另一方面，MEAN 14 不必做這麼多。它在翻譯時也較省事，並且您在給運算元時，可以指定指標的値或者直接提供浮點的數値，所以浮點常規不必去搜尋或轉換什麼。

MEAN 14 將時間花在數值的處理——而不是花時間去找數值所在或轉換 ASCII 碼。

MEAN 14 做些什麼

MEAN 14 是一個非常簡單的翻譯器。您給它一個數字，它會在表格中尋找所指定函數的副常規位址。大部份的函數都已存在 APPLESOFT 。有一些則需要再設立，以使得進入點較容易處理。在所有的努力下，我們設計一套指令集使得在使用浮點運算時較容易，但只能順序的執行指令。

上面提到了MEAN 14 的缺點，即只能順序的執行指令。目前在指令集內尚未包括條件指令。當您使用MEAN 14 時並不會那麼糟。在我自己使用時，我發現可以將測試、分歧、以及迴路動作放在MEAN 14 外面，以 6502 的組合語言來做。主要是因為所花的時間太多。基於這個理由，我們在MEAN 14 指令集中不包括條件指令。但如果您的程式需要的話，也可加上去。

MEAN 14的使用

在您的機器語言程式中使用MEAN 14 是非常容易的。唯一的要求是，除了要懂組合語言之外，也要知道APPLE-

SOFT 變數的格式。

1. 注意，召用MEAN 14和APPLESOFT副常規時，會使得所有暫存器的值處在不可決定的狀態下。如果在某一狀態下希望得到某一暫存器的值，則最好寫一個Save和一個Restore常規，然後在進入MEAN 14之前要JSR到Save常規。您可以隨您喜歡將這些常規加入MEAN 14中。
2. 利用一個 JSR 指令進入MEAN 14（即$8E00）。在這個 JSR 指令和MEAN 14的“ RET ”之間所有的碼就是MEAN 14處理機。記住，位元組的順序是位址型式（ addressing mode ）的函數。在內含型式（ Implied mode ），任何運算符號將接着一個運算符號。在即值型式（ Immediate mode），一個運算符號將接着一個APPLESOFT浮點變數格式的5位元組的運算元（常數）。在絕對型式（ Absolute mode），一個運算符號緊接着一個2個位元組的指標，此指標將指向一含有浮點數值的記憶體位置。在間接型式（ Indirect mode），運算符號緊接着一個指標，而此指標又指向另一指標，這指標才指向浮點數值。記住，所有的指標以標準 6502 的低一位元組，高一位元組次序爲主。
3. 看看下列指令：

```
2000 SUB1      STY YSAVE        ; SAVE Y
2002           STX XSAVE        ; SAVE X
2004           JSR MEAN 14      ; ENTER MEAN 14
2007           DFB C0 00 03     ; *LDA $300
200A           DFB C4 05 03     ; *ADD $305
200D           DFB 45 81 00
2010           DFB 00 00 00     ; *SUB #1
2013           DFB 0C           ; *ABS
2014           DFB 81 40 03     ; *STA ($340)
2017           DFB 11           ; *RET
2018           LDX XSAVE        ; RESTORE X
201A           LDY YSAVE        ; RESTORE Y
201C           RTS
```

例題中，在進入MEAN 14之前，X暫存器和Y暫存器都已存起來。爲了避免混淆，我們在說明欄說明MEAN 14助記憶符號 DFB 的意義。我將它們以星號開始爲記號。

4. 假使您的機器語言常規將被 BASIC 叫用使用，且 BASIC 將使用MEAN 14運算的結果，則您可能希望將此值直接送到記憶體中 APPLESOFT 變數所佔的位置上。這將使得 BASIC 能直接使用機器語言程式運算的結果。雖然在 APPLESOFT 中有副常規，能靠一個名字找到變數，但要花很多時間。最簡單的方法就是在您的 BASIC 程式中知道變數的位置所在。因此，如果您的程式第一行爲：

```
10  A = 0 : B = 0 : C = 0 : D = 0
```

則您將會知道第一個變數爲 A，第二個爲 B，等等。在

位置 $69、$69A 上的指標將告訴您變數存放的起始位置，所以其他均可得到。

5. 要小心避免浮點錯誤，例如滿溢現象（ overflow ）和除以零的運算，如果有錯誤發生，APPLESOFT 常規會將您傾印（ dump ）到 BASIC 中。

MEAN 14運算符號的格式

MEAN 14 的指令由一個位元組的值來表示。此位元組內共含二項意義——指令和位址型式。因為我們可以將位元組以簡單的結構包括此二項（只有 4 種位址型式與 20 個指令）。在此尚有未用的值，所以指令集可以再擴充。在一個位元組中，以前二個位元表示位址型式，剩下六個位元表示運算動作。

7 6	5 4 3 2 1 0
位址型式	指令

MEAN 14位址型式

MEAN 14 虛假處理機指令使用 4 種位址型式。它們是:

IMMEDIATE
ABSOLUTE
INDIRECT
IMPLIED

IMMEDIATE（即值型式）——像其他處理機一樣，MEAN 14的指令也允許使用即值型式，即在運算符號之後的運算元是數值。因爲我們是處理浮點數值，所以運算符號之後的5個記憶體位置必須包含浮點運算元。並且要以APPLESOFT 變數的格式表示。

例如，將“0”存入FPAC1

40	00 00 00 00 00	LDA#0
運算符號	運 算 元	符號

ABSOLUTE（絕對型式）——在指令的運算符號之後的二個位元組必定包含浮點數值第一個位元組的位址。指標所指的值，以及指標的值都是以低一位元組，高一位元組的格式存在。

例如，將FPAC1存在位址$1F00-$1F04

C1	00 1F	STA $1F00-$1F04
運算符號	運算元	符 號

INDIRECT（間接型式）——在這位址型式中，運算符號之後的二個位元組含着另一個二個位元組指標的位址，此指標將指向緩衝器的第一個位元組。在處理迴路時，此種位址型式很有用。它允許指向變數的指標改變，且因爲此指標不是MEAN 14機器碼的一部份，所以執行迴路時不必修改程式。運算元和指標也要以低一位元組、高一位元組格式表示。

例如，將FPAC1存在$2FF0-$2FF4

```
81    00 20    STA ($2000)
```

（其中$2000、$2001指向$2FF0）

IMPLIED（內含型式）——一些不包括變數的指令。包括暫存器函數，以及從MEAN 14返回。

例如，從FPAC1送到FPAC2

```
02    TAB
```

例如：從MEAN 14返回

```
11    RET
```

MEAN 14指令集

LDA	從記憶體載入FPAC1 IMMEDIATE = $40 ABSOLUTE = $C0 INDIRECT = $80	M→ FPAC1
STA	將FPAC1存入記憶體 ABSOLUTE = $C1 INDIRECT = $81	FPAC1 →M
TAB	將FPAC1的值送到 FPAC2 IMPLIED = $02	FPAC1 → FPAC2
TBA	將FPAC2的值送到 FPAC1 IMPLIED = $03	FPAC2 → FPAC1
ADD	將記憶體的值加到 FPAC1 IMMEDIATE = $44 ABSOLUTE = $C4 INDIRECT = $84	M + FPAC1 → FPAC1

SUB	將記憶體的值減 FPAC1，存入 FPAC1 IMMEDIATE = $45 ABSOLUTE = $C5 INDIRECT = $85	M − FPAC1 → FPAC1
MUL	將記憶體的值乘上 FPAC1 IMMEDIATE = $46 ABSOLUTE = $C6 INDIRECT = $86	M * FPAC1 → FPAC1
DIV	將記憶體的值除以 FPAC1 IMMEDIATE = $47 ABSOLUTE = $C7 INDIRECT = $87	M / FPAC1 → FPAC1
NOP	沒有動作 IMPLIED = $08	MPC + 1
SQR	FPAC1 的平方根 IMPLIED = $09	$\sqrt{FPAC1}$ → FPAC1

EXP	FPAC 2的記憶體值次方 IMMEDIATE = $4A ABSOLUTE = $CA INDIRECT = $8A	FPAC2 △M→ FPAC1
INT	取FPAC1的整數值 IMPLIED = $0B	INT (FPAC1)→ FPAC1
ABS	取FPAC1的絕對值 IMPLIED = $0C	ABS (FPAC1)→ FPAC1
SGN	取FPAC1的SIN函數值 IMPLIED = $0D	SGN (FPAC1)→ FPAC1
LOG	取FPAC1的自然對數 IMPLIED = $0E	LOG (FPAC1)→ FPAC1
CVA	將以APPLESOFT整數變數格式表示的二個位元組整數轉換成相對的浮點值 ABSOLUTE = $CF INDIRECT = $8F	M%→FPAC1

CVB	將以 6502 格式表示的二個位元組整數轉換成相對的浮點值 ABSOLUTE = $D0 INDIRECT = $90	ML、MH→FPAC1
RET	從MEAN 14 返回 IMPLIED = $11	MPC→PC

```
0800    ;*************************
0800    ;*                       *
0800    ;* MEAN-14 FP PROCESSOR  *
0800    ;*     R.M. MOTTOLA      *
0800    ;*                       *
0800    ;*        MEAN-14        *
0800    ;*                       *
0800    ;*   COPYRIGHT (C) 1981  *
0800    ;*    MICRO INK, INC.    *
0800    ;* CHELMSFORD, MA 01824  *
0800    ;*  ALL RIGHTS RESERVED  *
0800    ;*                       *
0800    ;*************************
0800    ;
0800    ;*SOFTWARE ADDRESSES
0800    ;*
0800    TEMPL   EPZ $1E
0800    TEMPH   EPZ $1F
0800    MPCL    EPZ $4C
0800    MPCH    EPZ $4D
0800    FPAC1   EPZ $9D
0800    FPAC2   EPZ $A5
0800    ;
0800    ;
0800    ;
0800    ;FIRMWARE ADDRESSES
0800    ;
0800    ;
0800    INT>FP  EQU $E2F2
0800    FPSUB   EQU $E7A7
0800    FPADD   EQU $E7BE
0800    FPLOG   EQU $E941
0800    FPMUL   EQU $E97F
0800    FPDIV1  EQU $EA66
```

```
0800              FPLOAD EQU $EAF9
0800              FPSTR  EQU $EB2B
0800              TR2>1  EQU $EB53
0800              TR1>2  EQU $EB63
0800              FPSGN  EQU $EB90
0800              FPABS  EQU $EBAF
0800              FPINT  EQU $EC23
0800              FPSQR  EQU $EE8D
0800              FPEXP  EQU $EE94
0800              ;
8E00                     ORG $8E00
8E00                     OBJ $800
8E00              ;
8E00              ;MEAN 14 PSEUDO-MACHINE
8E00              ;FLOATING POINT PROCESSOR
8E00              ;
8E00 68           MEAN14 PLA                ;GET M14 CODE LOCATION
8E01 854C                STA MPCL           ;FROM RETURN ADDRESS
8E03 68                  PLA
8E04 854D                STA MPCH
8E06 205F8E              JSR PCINC
8E09 200F8E       M14A   JSR M14B
8E0C 4C098E              JMP M14A
8E0F A000         M14B   LDY #$0
8E11 B14C                LDA (MPCL),Y       ;GET ONE INSTRUCTION
8E13 AA                  TAX
8E14 293F                AND #$3F           ;GET CORRECT SUBROUTINE
8E16 0A                  ASL                ;ADDRESS FROM TABLE
8E17 A8                  TAY
8E18 C8                  INY
8E19 B9A08E              LDA SUBTBL,Y       ;AND SHOVE IT
8E1C 48                  PHA
8E1D 88                  DEY
8E1E B9A08E              LDA SUBTBL,Y
8E21 48                  PHA
8E22 205F8E              JSR PCINC          ;INCREM. M14 P.C. COUNT
8E25 8A                  TXA
8E26 29C0                AND #$C0           ;GET ADDRESSING MODE
8E28 F034                BEQ M14G           ;IMPLIED?
8E2A 1020                BPL M14D           ;IMMEDIATE?
8E2C 2940                AND #$40
8E2E D013                BNE M14C           ;ABSOLUTE?
8E30 B14C                LDA (MPCL),Y       ;INDIRECT
8E32 851E                STA TEMPL          ;GET POINTER TO ADDRESS
8E34 C8                  INY                ;OF OPERAND
8E35 B14C                LDA (MPCL),Y
8E37 851F                STA TEMPH
8E39 88                  DEY
8E3A B11E                LDA (TEMPL),Y
8E3C 48                  PHA
8E3D C8                  INY
8E3E B11E                LDA (TEMPL),Y
8E40 48                  PHA
8E41 9013                BCC M14E
8E43 B14C         M14C   LDA (MPCL),Y       ;GET ADDRESS OF
8E45 48                  PHA                ;OPERAND
8E46 C8                  INY
8E47 B14C                LDA (MPCL),Y
8E49 48                  PHA
8E4A 900A                BCC M14E
8E4C A54C         M14D   LDA MPCL           ;SAVE P.C. AS ADDRESS
8E4E 48                  PHA                ;OF IMMEDIATE OPERAND
8E4F A54D                LDA MPCH
8E51 48                  PHA
```

```
8E52 A905              LDA #$5          ;AND OFFSET P.C. 5 BYTES
8E54 9002              BCC M14F
8E56 A902       M14E   LDA #$2          ;OFFSET P.C. 2 BYTES
8E58 20618E     M14F   JSR PCADD
8E5B 68                PLA              ;PULL OPERAND ADDRESS
8E5C            ;AND TRANSFER
8E5C A8                TAY              ;TO A AND Y REGS FOR SUBS
8E5D 68                PLA
8E5E 60         M14G   RTS              ;JMP VIA RTS
8E5F            ;
8E5F A901       PCINC  LDA #$1
8E61 18         PCADD  CLC
8E62 654C              ADC MPCL
8E64 854C              STA MPCL
8E66 9003              BCC PC1
8E68 E64D              INC MPCH
8E6A 18                CLC
8E6B A000       PC1    LDY #$0
8E6D 60                RTS
8E6E            ;
8E6E AA         STR    TAX
8E6F 4C2BEB            JMP FPSTR
8E72 851E       CONV1  STA TEMPL
8E74 841F              STY TEMPH
8E76 A000              LDY #$0
8E78 B11E              LDA (TEMPL),Y
8E7A 48                PHA
8E7B C8                INY
8E7C B11E       C1A    LDA (TEMPL),Y
8E7E A8                TAY
8E7F 68                PLA
8E80 20F2E2            JSR INT>FP
8E83 A5A2              LDA FPAC1+$5
8E85 1007              BPL NOOP
8E87 A9C4              LDA #VALUE1
8E89 A08E              LDY /VALUE1
8E8B 20BEE7            JSR FPADD
8E8E 60         NOOP   RTS
8E8F 851E       CONV2  STA TEMPL
8E91 841F              STY TEMPH
8E93 A001              LDY #$1
8E95 B11E              LDA (TEMPL),Y
8E97 48                PHA
8E98 88                DEY
8E99 F0E1              BEQ C1A
8E9B 68         RETURN PLA              ;PULL MEAN 14 RETURN
8E9C 68                PLA              ;ADDRESS FROM STACK
8E9D 6C4C00            JMP (MPCL)
8EA0            ;
8EA0            ;
8EA0            ;SUBROUTINE ADDRESS TABLE
8EA0            ;
8EA0 F6EA       SUBTBL ADR FPLOAD-$1
8EA2 6D8E              ADR STR-$1
8EA4 62EB              ADR TR1>2-$1
8EA6 52EB              ADR TR2>1-$1
8EA8 BDE7              ADR FPADD-$1
8EAA A6E7              ADR FPSUB-$1
8EAC 7EE9              ADR FPMUL-$1
8EAE 65EA              ADR FPDIV1-$1
8EB0 8D8E              ADR NOOP-$1
8EB2 8CEE              ADR FPSQR-$1
8EB4 93EE              ADR FPEXP-$1
8EB6 22EC              ADR FPINT-$1
```

```
8EB8 AEEB              ADR FPABS-$1
8EBA 8FEB              ADR FPSGN-$1
8EBC 40E9              ADR FPLOG-$1
8EBE 718E              ADR CONV1-$1
8EC0 8E8E              ADR CONV2-$1
8EC2 9A8E              ADR RETURN-$1
8EC4              ;
8EC4              ;FLOATING POINT CONSTANTS
8EC4              ;
8EC4 910000       VALUE1 HEX 9100000000       ; & 65536
8EC7 0000
8EC9              ;
8EC9              ;
8EC9              ;
8EC9              ;
8EC9              LENGTH EQU *-MEAN14
                         END

***** END OF ASSEMBLY
```

```
          **************************
          *                        *
          * SYMBOL TABLE -- V 1.5  *
          *                        *
          **************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

TEMPL  001E  TEMPH  001F  MPCL   004C  MPCH   004D  FPAC1  009D  FPAC2  00A5

** ABSOLUTE VARABLES/LABELS

INT>FP E2F2  FPSUB  E7A7  FPADD  E7BE  FPLOG  E941  FPMUL  E97F  FPDIV1 EA66
FPLOAD EAF9  FPSTR  EB2B  TR2>1  EB53  TR1>2  EB63  FPSGN  EB90  FPABS  EBAF
FPINT  EC23  FPSQR  EE8D  FPEXP  EE94  MEAN14 8E00  M14A   8E09  M14B   8E0F
M14C   8E43  M14D   8E4C  M14E   8E56  M14F   8E58  M14G   8E5E  PCINC  8E5F
PCADD  8E61  PC1    8E6B  STR    8E6E  CONV1  8E72  C1A    8E7C  NOOP   8E8E
CONV2  8E8F  RETURN 8E9B  SUBTBL 8EA0  VALUE1 8EC4  LENGTH 00C9

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:015A
```

2 加强輸入／輸出功能

簡 介

螢幕的寫入／存檔常規
——— B. E. Baxter

雙向自動捲頁
——— Roger Wagner

一頁一頁的列印 APPLE II 整數 BASIC 程式
——— Dave Partyka

APPLE 印表機的編頁輸出
——— Gary Little

十六進位印表機
——— Leroy Moyer

第二部份簡介

為了和您的電腦互通訊息，必須要有一個輸入／輸出裝置。鍵盤輸入和電視輸出是APPLE的標準輸入／輸出裝置，印表機也是一個非常普遍的輸出裝置。明顯的，加強輸入／輸出的能力將使得使用者和電腦之間的介面變得更好。在本章，我們將介紹一些改進方法，使您在使用APPLE電腦時更容易。

由Bruce Baxter寫的“螢幕的寫入／存檔常規”，提供一個方法直接將正文螢幕存入磁碟中，並從磁碟中取出正文資料。這種技巧能節省大量的記憶體空間。Roger Wagner的“雙向自動捲頁”，允許在記憶體中往前或往後自動捲頁。記憶體的任何部份均可經由此常規顯示ASCII碼於螢幕上。David Partyka 的“一頁一頁的列印APPLE II 整數 BASIC程式”，允許使用者在APPLE的螢幕上一頁一頁的列印整數 BASIC 程式。

下面的二個常規只針對擁有印表機的使用者。Gray Little的“APPLE印表機的跳頁輸出”，允許印表機將輸

出分成好幾頁印出，每頁的大小可以調整。並且允許在印完一頁之後暫時停止輸出。LeRoy Moyer的“十六進位印表機”，使您的印表機能列印出解組合（disassembly）的機器語言程式。

螢幕的寫入/存檔常規

作者： B. E. Baxter

這是一個非常有用的常規，使您在編輯APPLE 螢幕時更簡單，以及將螢幕的影像存入磁碟中。

此常規只有 73 個位元組，它將由監督系統取得控制權，並且直接寫入螢幕。它提供所有的 Escape 編輯指令，所以很容易的能修改螢幕上的 21 行資料。修改完螢幕上的資料後，也可很容易將螢幕上的影像存入磁碟中。

它如何工作

此常規用到許多監督系統常規。開始時，游標（cursor）出現在螢幕的左上方，即第一行。在程式中標記爲 KEY 的方塊將不斷的詢問鍵盤是否有資料輸入，並且經由 COUT 輸出資料（如果不用印表機，則可用VIDOUT[$FBFD]）。它能使用監督系統的部份編輯指令，只要按 Escape 鍵，再按命令符號。從鍵盤按 CNTL Q 將停止執行此常規，且

經由 $3D0 回到 BASIC。在程式的 43 行會自動停止執行。停止執行時，將會有嗶嗶的聲音，並且會出現 BASIC 的提示符號，最後一行會顯示檔案的參數。此時若要將檔案存入磁碟，要用（BSAVE 檔案名稱）A$0400、L$03CF（Return）。括號內的字是使用者要打入的；即打入（BSAVE 檔案名稱），就會追踪剩餘的行數，並正確的放入鍵盤的緩衝器中，在最後一行再打入（Return）。我們也可利用監督系統的 MOVE 指述移到第 2 頁，且將其加於程式 57 行和 58 行之間，但我認爲這不一定要。只有在 BSAVE 不能動作時，它可以提供一個補救的方法。程式由 $350 位址開始，共佔 $49 個位元組。

命令摘要

進入常規

從 BASIC	CALL 848
從監督系統	$0350G

結束回到 BASIC

使　用　者	（Control）Q
自　　動	43 行

螢幕編輯

（Escape）	@：游標回到左上方（清除螢幕資料）

A：游標往前進 1 格

B：游標往後退 1 格

C：游標往下跳 1 行

D：游標往上跳 1 行

E：從游標位置開始到此行最後一格的所有資料都清除掉。

F：從游標位置開始到螢幕最後一行的所有資料都清除掉。

將螢幕影像存起來

[BSAVE 檔案名稱] A$0400 , L$03CF

[Return]

當然，若沒有任何目的而隨意在螢幕上寫是沒有意義的。我使用此常規來產生指令檔案與文件檔案。在處理公用程式的使用法說明與進入點消息時，這些檔案很有用。一旦這些檔案產生後，它可以像其他檔案一樣處理。打入（BLOAD 檔案名稱）將會把檔案的內容顯示在螢幕上而不需要其他有用的記憶體。所以 BASIC 的 PRINT 指述便可省略，而以執行磁碟命令 BLOAD 取代，將會有效的使用記憶體。

```
0800        ;*************************
0800        ;*                       *
0800        ;*    SCREEN WRITER      *
0800        ;*     BRUCE BAXTER      *
0800        ;*                       *
0800        ;*     SCREEN-WRITE      *
0800        ;*                       *
0800        ;*  COPYRIGHT (C) 1981   *
0800        ;*   MICRO INK, INC.     *
```

```
0800               ;* CHELMSFORD, MA 01824 *
0800               ;*  ALL RIGHTS RESERVED *
0800               ;*                      *
0800               ;************************
0800               ;
0800               ;
0350                       ORG $350
0350                       OBJ $800
0350               ;
0350               ;
0350               ;
0350               CV      EPZ $25
0350               POS     EPZ $09
0350               ;
0350               COUT    EQU $FDED
0350               HOME    EQU $FC58
0350               TABV    EQU $FB5B
0350               RDCHAR  EQU $FD35
0350               CROUT   EQU $FD8E
0350               BELL    EQU $FF3A
0350               ;
0350 2058FC                JSR HOME
0353 208EFD                JSR CROUT
0356               ;
0356 2035FD        KEY     JSR RDCHAR
0359 C991                  CMP #$91
035B F00C                  BEQ QUIT
035D A625                  LDX CV
035F E016                  CPX #$16
0361 F006                  BEQ QUIT
0363 20EDFD                JSR COUT
0366 4C5603                JMP KEY
0369               ;
0369 A916          QUIT    LDA #$16
036B 8525                  STA CV
036D 205BFB                JSR TABV
0370 203AFF                JSR BELL
0373 A9E4                  LDA #$E4
0375 8509                  STA POS
0377 A907                  LDA #$07
0379 850A                  STA POS+1
037B A000                  LDY #$00
037D               ;
037D B98A03        OUT     LDA DATA,Y
038C 9109                  STA (POS),Y
0382 C8                    INY
0383 C00F                  CPY #$0F
0385 D0F6                  BNE OUT
0387 20D003                JSR $03D0
038A A0C1A4        DATA    ASC " A$C400,L$03CF "
038D B0B4B0
0390 B0ACCC
0393 A4B0B3
0396 C3C6A0
                           END

***** END OF ASSEMBLY
```

```
          *************************
          *                       *
          * SYMBOL TABLE -- V 1.5 *
          *                       *
          *************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

CV     0025  POS    0009

** ABSOLUTE VARABLES/LABELS

CCUT   FDED  HOME   FC58  TABV   FB5B  RDCHAR FD35
CROUT  FD8E  BELL   FF3A  KEY    0356  QUIT   0369  OUT    037D  DATA   038A


SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0072
```

雙向自動捲頁

作者：Roger Wagner

每個人都知道電傳打字機上的紙其移動方向只有一個——往上，而APPLE 在自動捲頁顯示時也只有一個方向——往上。現在您可以利用以下的常規往二個方向自動捲頁顯示。

利用下列的機器語言常規，能夠往二個方向掃瞄文字幕或圖形資料。

往上捲頁的常規是參考APPLE 電腦參考手册，不同的是參考第0頁的位置以便決定那一頁要往上捲。往下捲頁的常規也是對第0頁做類似的動作。

如何使用此程式

使用前必須符合下列條件：

1. 將此常規載入到記憶體中，從$300開始。

2. 設立指標 6 ， 7 和 8 ， 9 。當您在自動捲頁時，如果希望加入一些新的資訊，則把指標 6 ， 7 指向這些資料的位置，當您在螢幕上往下捲頁時，這些資料便會從第一行開始顯示在螢幕上。同理的，指標 8 ， 9 所指的資料，在往上捲頁時會出現在螢幕的底端。

假如您想利用此常規直接看記憶體內的資料，最簡單的方法就是設立指標 6 ， 7 和 8 ， 9 的值，首先設立指標 8 ， 9 為所要看的起始位置。指標 8 內存低一位元組、指標 9 內存高一位元組，並往上捲頁 25 次（螢幕的高度加 1 ）。接着設立指標 6 ， 7 和指標 8 ， 9 原來的值一樣（低一位元組和高一位元組表示起始位置）。最後，由起始位置往下捲頁一行，就如在螢幕上所看到的第一行資料。

如果您不想在螢幕上加入新的資料，則指標 6 ， 7 和 8 ， 9 所指的位置一定要是含有 40 個空白字元的記憶體位置。一種方法就是在第 1 頁或第 2 頁記憶體中保留一空白行，然後令指標 6 ， 7 和指標 8 ， 9 指向此位置。當每次捲頁結束後，指標的值必須重新指向此位置。這是因為捲頁常規利用螢幕的寬度來更改指標 6 ， 7 和 8 ， 9 的值，以便在螢幕顯示時能保持同步。另一種技巧是，每當捲頁結束時，便將最上面或最底端的一行清除為空白行。

3. 在捲第 1 頁時，位址 5 的值為 4 ，在捲第 2 頁時，位址 5 的值為 8 。

4. 現在如果想要螢幕自動捲頁，則打入“CALL 768”便可往上捲頁，打入“CALL 845”便可往下捲頁。

注意事項

假如您在整數 BASIC 中想使用第2頁的文字幕或圖形資料，記着要用“LOMEM”：3072 保護變數的值。這可在執行程式之前做好；或者您知道如何做，可放在程式的前幾行。

假如要在 APPLESOFT 中使用第2頁記憶體則比較困難，但還是可以做。首先，在位址 $3AB 中的值 $05 要改成 $1F。同樣的，您必須利用 POKE 指述將位址 31 的值改為4或8，就如整數BASIC改變位址5一樣。

最大的障礙就是，APPLESOFT 程式通常都由位址 $800 開始，而這將和使用第2頁記憶體衝突。避免衝突的方法就是在載入程式之前執行“POKE 104,12：POKE 3072,0”。載入之後再打入“CALL 54514”（使用 DOS 3.2 則不必）。除非您按下“RESET”、“Control－B”，否則其他APPLESOFT 程式會繼續載入到較高的位址上。很不幸的，在 RAM 型式的 APPLESOFT 要使用第2頁記憶體，以我所知道的是不可能做到（對不起……）。

假如您希望將捲頁常規移到記憶體的另一位址，只有更

改5個“ JSR ”和1個“ JMP ”的碼，這些都是和位址有關的指令。因爲它們所參考的都是絕對位址，所以必須要重寫。當然，如果您有重新定位址的常規，這會更容易做到。

爲了更進一步的了解，請看使用捲頁常規的整數 BASIC 程式範例。祝愉快！

和位址有關的指令：

```
$303 : JSR  $39E
 319 : JSR   39E
 34A : JMP   39C
 353 : JSR   39E
 369 : JSR   39E
 39E : JSR   3A6
```

如果要使用第2頁記憶體，對整數 BASIC 要用“ LOMEM： 3072，保護；對 APPLESOFT 要用一個“特定的載入動作”（如本文所提）。

注意：對 APPLESOFT 而言，位址 $3AB 所含的值，即$05，要改成$1F。

```
0800    ;*****************************
0800    ;*                           *
0800    ;* APPLE SCROLLING ROUTINE   *
0800    ;*       ROGER WAGNER        *
0800    ;*                           *
0800    ;*          SCROLL           *
0800    ;*                           *
0800    ;*   COPYRIGHT (C) 1981      *
```

```
0800                ;*     MICRO INK, INC.         *
0800                ;*   CHELMSFORD, MA 01824      *
0800                ;*    ALL RIGHTS RESERVED      *
0800                ;*                             *
0800                ;*******************************
0800                ;
0800                ; THIS WILL LET EITHER
0800                ; PAGE SCROLL IN EITHER
0800                ; DIRECTION.  IT IS PRI-
0800                ; MARILY DESIGNED TO FEED
0800                ; NEW SCREEN DATA IN FROM
0800                ; A GIVEN RANGE OF RAM.
0800                ;
0800                ;
0800                ;
0800                       OBJ $800
0300                       ORG $300
0300                ;
0300                ;
0300                WNDLFT EPZ $20
0300                WNDWID EPZ $21
0300                WNDTOP EPZ $22
0300                WNDBTM EPZ $23
0300                CH     EPZ $24
0300                CV     EPZ $25
0300                BASL   EPZ $28
0300                BASH   EPZ $29
0300                BAS2L  EPZ $2A
0300                BAS2H  EPZ $2B
0300                PAGE   EPZ $05
0300                ;* FOR APPLESOFT USE PAGE EQU $1F
0300                ;* PAGE MUST HOLD $04 FOR PG 1,
0300                ;* $08 FOR PG 2
0300                SCRNTP EPZ $06
0300                ;* $06, $07 = LO/HI BYTES
0300                ;* OF START OF LINE JUST BEFORE
0300                ;* TOP LINE
0300                SCRNBM EPZ $08
0300                ;* $08,$09 = LO/HI BYTES
0300                ;* OF START OF LINE JUST AFTER
0300                ;* BOTTOM LINE
0300                ;*
0300                ;*
0300 A522           SCROLL LDA WNDTOP
0302 48                    PHA
0303 209E03                JSR VTABZ
0306 A528           NXTLN  LDA BASL
0308 852A                  STA BAS2L
030A A529                  LDA BASH
030C 852B                  STA BAS2H
030E A421                  LDY WNDWID
0310 88                    DEY
0311 68                    PLA
0312 6901                  ADC #$01
0314 C523                  CMP WNDBTM
0316 B00D                  BCS LDBTM
0318 48                    PHA
```

```
0319 209EC3          JSR VTABZ
031C B128    NXTCHR  LDA (BASL),Y
031E 912A            STA (BAS2L),Y
0320 88              DEY
0321 10F9            BPL NXTCHR
0323 30E1            BMI NXTLN
0325 A000    LDBTM   LDY #00
0327 B108    LD2     LDA (SCRNBM),Y
0329 9128            STA (BASL),Y
032B C8              INY
032C C421            CPY WNDWID
032E 90F7            BCC LD2
0330 18      CRRCT   CLC
0331 A506            LDA SCRNTP
0333 6521            ADC WNDWID
0335 8506            STA SCRNTP
0337 A507            LDA SCRNTP+1
0339 6900            ADC #00
033B 8507            STA SCRNTP+1
033D 18              CLC
033E A508            LDA SCRNBM
0340 6521            ADC WNDWID
0342 8508            STA SCRNBM
0344 A509            LDA SCRNBM+1
0346 6900            ADC #00
0348 8509            STA SCRNBM+1
034A 4C9C03          JMP VTAB
034D         ;*
034D         ;*
034D 38      SCRLDN  SEC
034E A523            LDA WNDBTM
0350 E901            SBC #$01
0352 48              PHA
0353 209E03          JSR VTABZ
0356 A528    NXTLN2  LDA BASL
0358 852A            STA BAS2L
035A A529            LDA BASH
035C 852B            STA BAS2H
035E A421            LDY WNDWID
0360 88              DEY
0361 68              PLA
0362 E900            SBC #$00
0364 C522            CMP WNDTOP
0366 30CD            BMI LDTOP
0368 48              PHA
0369 209E03          JSR VTABZ
036C B128    NXTCR2  LDA (BASL),Y
036E 912A            STA (BAS2L),Y
0370 88              DEY
0371 10F9            BPL NXTCR2
0373 30E1            BMI NXTLN2
0375 A000    LDTOP   LDY #$00
0377 B106    LT2     LDA (SCRNTP),Y
0379 9128            STA (BASL),Y
037B C8              INY
037C C421            CPY WNDWID
037E 90F7            BCC LT2
0380 38      CRRT2   SEC
```

```
0381 A506              LDA SCRNTP
0383 E521              SBC WNDWID
0385 8506              STA SCRNTP
0387 A507              LDA SCRNTP+1
0389 E900              SBC #00
038B 8507              STA SCRNTP+1
038D 38                SEC
038E A508              LDA SCRNBM
0390 E521              SBC WNDWID
0392 8508              STA SCRNBM
0394 A509              LDA SCRNBM+1
0396 E900              SBC #00
0398 8509              STA SCRNBM+1
039A 60                RTS
039B 00                BRK
039C              ;*
039C              ;*
039C A525         VTAB    LDA CV
039E 20A603       VTABZ   JSR BASCLC
03A1 6520              ADC WNDLFT
03A3 8528              STA BASL
03A5 60                RTS
03A6              ;*
03A6              ;*
03A6 48           BASCLC  PHA
03A7 4A                LSR
03A8 2900              AND #$00
03AA 0505              ORA PAGE
03AC 8529              STA BASH
03AE 68                PLA
03AF 2918              AND #$18
03B1 9002              BCC BSCLC2
03B3 697F              ADC #$7F
03B5 8528         BSCLC2  STA BASL
03B7 0A                ASL
03B8 0A                ASL
03B9 0528              ORA BASL
03BB 8528              STA BASL
03BD 60           END     RTS
                       END

***** END OF ASSEMBLY


          ***************************
          *                         *
          * SYMBOL TABLE -- V 1.5   *
          *                         *
          ***************************

LABEL. LOC.   LABEL. LOC.   LABEL. LOC.

** ZERO PAGE VARIABLES:

WNDLFT 0020  WNDWID 0021  WNDTOP 0022 WNDBTM 0023 CH     0024 CV     0025
BASL   0028  BASH   0029  BAS2L  002A BAS2H  002B PAGE   0005 SCRNTP 0006
SCRNBM 0008
```

```
** ABSOLUTE VARABLES/LABELS

SCROLL 0300   NXTLN  0306  NXTCHR 031C LDBTM  0325 LD2    0327
CRRCT  0330   SCRLDN 034D  NXTLN2 0356 NXTCR2 036C LDTOP  0375 LT2    0377
CRRT2  0380   VTAB   039C  VTABZ  039E BASCLC 03A6 BSCLC2 03B5 END    03BD


SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:01C2
  1 REM    **************************
  2 REM    *                        *
  3 REM    * APPLE SCROLLING RTNE   *
  4 REM    *     ROGER WAGNER       *
  5 REM    *                        *
  6 REM    *       SCROLLER         *
  7 REM    *                        *
  8 REM    *  COPYRIGHT (C) 1981    *
  9 REM    *    MICRO INK, INC.     *
 10 REM    * CHELMSFORD, MA 01824   *
 11 REM    *  ALL RIGHTS RESERVED   *
 12 REM    *                        *
 13 REM    **************************
 14 REM
 16 REM
 18 LOMEM:3072
 20 REM    OR SET LOMEM MANUALLY BEFORE RUNNING.
 30 CALL -936: INPUT "PAGE 1 OR 2?",PAGE
 40 PRINT "INPUT ADDRESS (<32767) TO START AT:": INPUT A
 50 REM    TO SCROLL WITHOUT BRINGING IN NEW DATA ENTER '0' FOR ADDRESS.
 60 IF A#0 THEN 100: TEXT : CALL -936: POKE 34,1:
    REM    FREEZE ONE BLANK LINE AT TOP OF SCREEN
 70 VTAB 12: PRINT "(SAMPLE PG. 1 SCREEN DATA)"
 75 POKE 5,PAGE*4: IF PAGE=2 THEN POKE -16299,0
 80 POKE 6,0: POKE 7,4: POKE 8,0: POKE 9,4:
    REM    BRING NEW SCREEN DATA FROM  THAT BLANK LINE
 90 GOTO 150
100 LB=A MOD 256:HB=A/256
110 POKE 5,PAGE*4: IF PAGE=2 THEN POKE -16299,0
120 POKE 8,LB: POKE 9,HB
130 FOR I=1 TO 25: CALL 768: NEXT I
140 POKE 6,LB: POKE 7,HB
150 KEY= PEEK (-16384): POKE -16368,0
160 IF KEY=149 THEN CALL 768: REM    RT. ARROW KEY TO SCROLL UP
170 IF KEY=136 THEN CALL 845: REM    LFT. ARROW KEY TO SCROLL DOWN
180 IF KEY#136 AND KEY#149 OR A#0 THEN 190: POKE 6,0: POKE 7,4: POKE 8,
    0: POKE 9,4: REM    RESET 6,7 & 8,9 TO POINT AT BLANK LINE
190 IF KEY#177 THEN 200: POKE 5,4: POKE -16300,0: REM    '1' FOR PAGE 1
200 IF KEY#178 THEN 210: POKE 5,8: POKE -16299,0: REM    '2' FOR PAGE 2
210 IF KEY#216 THEN 150: POKE -16300,0: TEXT : CALL -868: PRINT "BYE":
    END
```

一頁一頁的列印APPLE Ⅱ 整數BASIC程式

作者： Dave Partyka

在APPLE的小螢幕上看很長的程式對使用者來說，是一件很困擾的事。解決此問題的方法就是允許使用者能一頁一頁的查看印出的程式。

假如您擁有一部APPLE II，您將會發現在列印程式時，這是一個很好的方法。現在您只可以列印整個程式，但是它的速度却比您所能讀的速度還快，或者您可以利用行數號碼來列印。當您利用行數號碼列印程式時，您可以列印二行或者列印比您的螢幕所能容納的行數還多。

利用下列的組合語言程式，與APPLE II的整數BASIC，您可以一頁一頁（螢幕的大小）的列印您的整數BASIC 程式，並且在最底端顯示頁數。按下鍵盤上的任何一個鍵（除了B、P、或S之外），將會清除螢幕上的資料，並且把下一頁的資料顯示在螢幕上，頁數號碼也會加1。按一個鍵一次可以顯示一頁的程式，而不是只有二行或太多行以至螢幕容納不下。

B、P、或S鍵是一些特殊功能鍵。按下B鍵（從頭開始）將會清除螢幕資料，並且從程式的第一頁開始顯示。當

您正在顯示程式的中間或後面部份，而您想再顯示前面的一部份程式，利用此功能鍵是很方便的。只要按下B鍵，您就可以從頭開始顯示程式。

下一個功能鍵是P鍵（印第幾頁的意思），它會清除螢幕資料並從頭開始顯示程式，然後停在您所想要看的頁數上。例如，您現在正在列印第 25 頁，而您想往回看2頁，您可以打入P0023 。P將會清除螢幕，且當您打入那四個數字時，APPLE 會嗶一聲。在此您必須打入 4 個數字，前頭零（ leading zero ）也要打入。當打入最後一個數字元時，您的程式將會從頭開始顯示，而停在第 23 頁。這要比按B鍵，然後一頁一頁的列印到第 23 頁還快。

最後一個功能鍵S（停止），將使您由列印程式跳出來而回到APPLE II BASIC 。當您在程式的某位置發現要加入或消掉某行時，您就要使用此鍵。如果您不按此鍵，而想做其他的事情時，那是行不通的，因爲您每按一個鍵，下一頁程式將會顯示在螢幕上。

啓動此程式的方法有二種。從監督系統，先按 CTRL－Y，再按 RETURN 鍵，或者從BASIC，打入CALL 1016，再按 RETURN 鍵。只要您不用記憶體 $300 到 $3FF 的位置，這個程式將保存在記憶體中。一旦程式啓動後，必須在螢幕顯示到底端後才能打入指令。如果程式的最後指令不是顯示在螢幕的底端，則APPLE II將會回到 BASIC，不過列印程式仍在啓動狀態。若要停止列印程式，則打入CALL 1016 ，按RETURN 鍵，再按S鍵，或者按

RETURN 鍵跳到此頁的最底端，再按S鍵。

如果您在執行 BASIC 程式時，列印程式還在啓動狀態，則所得到的結果將依您的程式而定。有一些程式根本不會影響。有一些程式則當列印結果碰到螢幕的底端時就會停止，再按一個鍵就會再繼續執行。有一些程式可能利用此常規，在顯示結果碰到螢幕底端時就停止列印。

使用此組合語言常規，您將會發現很容易對您的程式除錯，或對照任何程式的流程。

```
0800            ;*************************
0800            ;*                       *
0800            ;*     LIST BY PAGE      *
0800            ;*     DAVID PARTYKA     *
0800            ;*                       *
0800            ;*       PAGE LIST       *
0800            ;*                       *
0800            ;*  COPYRIGHT (C) 1981   *
0800            ;*   MICRO INK, INC.     *
0800            ;* CHELMSFORD, MA 01824  *
0800            ;*  ALL RIGHTS RESERVED  *
0800            ;*                       *
0800            ;*************************
0800            ;
0800            ;
0800            ;
0800            BASL   EPZ $28          ;LEFT CHAR POS ON LINE
0800            BASH   EPZ $29
0800            CSWL   EPZ $36          ;MONITOR OUTPUT  HOOK
0800            CSWH   EPZ $37
0800            ;
0800            KBD    EQU $C000        ;KEYBOARD INPUT
0800            KBDSTB EQU $C010        ;KEYBOARD STROBE
0800            BASIC2 EQU $E003        ;BASIC WARM ENTRY
0800            LIST   EQU $E04B        ;BASIC LISTING ROUTINE
0800            BELL   EQU $FBDD        ;MONITOR BELL ROUTINE
0800            HOME   EQU $FC58        ;MONITOR CLEARSCREEN
0800            COUT1  EQU $FDF0        ;CHARACTER OUTPUT ROUTINE
0800            SAVE   EQU $FF4A        ;REGISTER SAVE ROUTINE
0800            RESTOR EQU $FF3F        ;REGISTER RESTORE ROUTINE
0800            ;
0300                   ORG $0300
0300                   OBJ $0800
0300            ;
0300            ;
0300 A922       INIT   LDA #MAIN        ;LOAD BEGINNING
```

```
0302 8536              STA CSWL          ;ADDRESS OF MAIN
0304 A903              LDA /MAIN         ;PROGRAM IN USER
0306 8537              STA CSWH          ;OUTPUT LOCATIONS.
0308 20E603    BEG     JSR HLOD          ; LOAD HIGH VALUES.
030B A900      ZPNO    LDA #$00          ;MOVE ZEROS TO
030D 8DF403            STA PGHI          ;PAGE COUNT
0310 8DF503            STA PGLO          ;LOCATIONS.
0313 2058FC            JSR HOME          ;CLEAR SCREEN.
0316 204BE0            JSR LIST          ;START BASIC LIST.
0319
0319 209603            JSR ADD1          ;ADD1 TO PAGE#.
031C 20E603            JSR HLOD          ;LOAD PAGE HOLD WITH FF.
031F 4C03E0            JMP BASIC2        ;RETURN TO BASIC CONTROL.
0322 204AFF    MAIN    JSR SAVE          ;SAVE REGISTERS
0325 A528              LDA BASL          ;CHECK SCREEN ADDRESS
0327 4529              EOR BASH          ;FOR 07 DO THE
0329 C9D7              CMP #$D7          ;24TH LINE.
032B D051              BNE DISP          ;IF NOT = BRANCH.
032D 209603            JSR ADD1          ;ADD 1 TO PAGE #.
0330
0330 ADF603            LDA PHOLD         ;CHECK PAGE HOLD,
0333 C9FF              CMP #$FF          ;IF = FF THEN THE P
0335 F019              BEQ NPRES         ; KEY WASN'T PRESSED.
0337 ADF403            LDA PGHI          ;COMPARE PAGE #
033A CDF603            CMP'PHOLD         ;WITH PAGE HOLD,
033D D008              BNE CLR           ;IF EQUAL
033F ADF503            LDA PGLO          ;BRANCH TO THE
0342 CDF703            CMP PHOLD+1       ;LOOP ROUTINE
0345 F006              BEQ LOOPR         ;ELSE
0347 2058FC    CLR     JSR HOME          ;CLEAR SCREEN
034A 4C8103            JMP RR            ;CONTINUE PRINTING.
034D 20E603    LOOPR   JSR HLOD          ;LOAD PAGE HOLD WITH FF.
0350 2C00C0    NPRES   BIT KBD           ;LOOP UNTIL A
0353 10FB              BPL NPRES         ;KEY IS PRESSED.
0355 AD00C0            LDA KBD           ;WHEN KEY IS PRESSED
0358 8D10C0            STA KBDSTB        ;CLEAR KEY STROBE
035B C9D3              CMP #$D3          ;AND COMPARE FOR S.
035D D0CB              BNE CMPB          ;IF NOT = BRANCH.
035F A9F0              LDA #$F0          ;IF S STORE
0361 8536              STA CSWL          ;NORMAL ADDRESS
0363
0363 A9FD              LDA #$FD          ;IN THE USER
0365 8537              STA CSWH          ;OUTPUT LOCATIONS.
0367 4C03E0            JMP BASIC2        ;RETURN TO BASIC CONTROL
036A C9C2      CMPB    CMP #$C2          ;B KEY PRESSED?
036C F09A              BEQ BEG           ;IF YES BRANCH.
036E C9D0              CMP #$D0          ;P KEY PRESSED?
0370 D00C              BNE DISP          ;IF NO BRANCH.
0372 A200              LDX #$00          ;IF YES THEN GET
0374 20CF03            JSR GTPG          ;2 DIGITS OF PAGE#
0377 E8                INX               ;UP INDEX AND
0378 20D203            JSR GTPG1         ;GET NEXT TWO DIGITS.
037B 4C0B03            JMP ZPNO          ;JUMP TO ZERO PAGE #.
037E 2058FC    DISP    JSR HOME          ;CLEAR SCREEN.
0381 203FFF    RR      JSR RESTOR        ;RESTORE REGISTERS
0384 4CF0FD            JMP COUT1         ;DISPLAY ROUTINE.
0387 A8        PRINT   TAY               ;SAVE ACCUM. AND
```

```
0388 290F              AND #$0F         ;CONVERT LOW ORDER
038A 09B0              ORA #$B0         ;BYTE TO DECIMAL AND
038C 9DF407            STA $7F4,X       ;PRINT PAGE #.
038F 98                TYA              ;GET ACCUM. AND
0390 6A                ROR              ;ROTATE
0391 6A                ROR              ;HIGH ORDER
0392 6A                ROR              ;BYTE TO THE
0393 6A                ROR              ;LOW ORDER
0394 CA                DEX              ;BYTE AND
0395 60                RTS              ;RETURN.
0396 F8         ADD1   SED              ;SET DECIMAL MODE.
0397 18                CLC              ;CLEAR CARRY FLAG.
0398 ADF503            LDA PGLO         ;ADD
039B 6901              ADC #$01         ;1
039D 8DF503            STA PGLO         ;TO
03A0 ADF403            LDA PGHI         ;THE
03A3 6900              ADC #$00         ;PAGE
03A5 8DF403            STA PGHI         ;NUMBER.
03A8 D8                CLD              ;CLEAR DECIMAL MODE.
03A9 A203              LDX #$03         ;SET IND-X.
03AB ADF503            LDA PGLO         ;GET PAGE # LOW.
03AE 208703            JSR PRINT        ;PRINT 1ST DIGIT.
03B1 208703            JSR PRINT        ;PRINT 2ND DIGIT.
03B4 ADF403            LDA PGHI         ;GET PAGE # HIGH.
03B7 208703            JSR PRINT        ;PRINT 3RD DIGIT.
03BA 208703            JSR PRINT        ;PRINT 4TH DIGIT.
03BD 60                RTS              ;RETURN.
03BE 2C00C0     KEY    BIT KBD          ;LOOP UNTIL A
03C1 10FB              BPL KEY          ;KEY IS PRESSED.
03C3 20DDFB            JSR BELL         ;RING BELL
03C6 AD00C0            LDA KBD          ;GET KEY
03C9 8D10C0            STA KBDSTB       ;CLEAR STROBE
03CC 290F              AND #$0F         ;DROP HIGH ORDER
03CE 60                RTS              ;HALF AND RETURN.
03CF 2058FC     GTPG   JSR HOME         ;CLEAR SCREEN.
03D2            ;
03D2 20BE03     GTPGI  JSR KEY          ;GET PAGE #.
03D5 0A                ASL              ;SHIFT LOW ORDER
03D6 0A                ASL              ;HALF TO THE
03D7 0A                ASL              ;HIGH ORDER
03D8 0A                ASL              ;HALF.
03D9 9DF603            STA PHOLD,X      ;STORE IN PAGE HOLD.
03DC 20BE03            JSR KEY          ;GET NEXT NUMBER.
03DF 5DF603            EOR PHOLD,X      ;COMBINE WITH
03E2 9DF603            STA PHOLD,X      ;PREVIOUS # AND STORE
03E5 60                RTS              ;IN PAGE HOLD, RETURN.
03E6            ;
03E6 A9FF       HLOD   LDA #$FF         ;PUT HIGH VALUES
03E8 8DF603            STA PHOLD        ;IN PAGE HOLD
03EB 8DF703            STA PHOLD+1      ;LOCATIONS THEN
03EE 60                RTS              ;RETURN.
03EF            ;
03EF 000000            HEX 0000000000
03F2 0000
03F4 00         PGHI   HEX 00           ;PAGE # HIGH
03F5 00         PGLO   HEX 00           ;PAGE # LOW
03F6 0000       PHOLD  HEX 0000         ;PAGE HOLD
03F8            ;
03F8 4C0003     CTRLY  JMP INIT
                       END
```

```
***** END OF ASSEMBLY


          *************************
          *                       *
          * SYMBOL TABLE -- V 1.5 *
          *                       *
          *************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

BASL    0028  BASH    0029  CSWL    0036  CSWH    0037

** ABSOLUTE VARABLES/LABELS

KBD     C000  KBDSTB C010
BASIC2 E003  LIST    E04B  BELL    FBDD  HOME  FC58 COUT1 FDF0  SAVE   FF4A
RESTOR FF3F  INIT    0300  BEG     0308  ZPNC  030B MAIN  0322  CLR    0347
LOOPR  034D  NPRES   0350  CMPB    036A  DISP  037E RR    0381  PRINT  0387
ADD1   0396  KEY     03BE  GTPG    03CF  GTPG1 03D2 HLOD  03E6  PGHI   03F4
PGLO   03F5  PHOLD   03F6  CTRLY   03F8

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:011A
```

APPLE印表機的編頁輸出

作者： Gary Little

在您的系統中加入編頁的模式，使您的輸出報表格式更好看。

假如您曾經從印表機印出結果，您會發現印出的結果從某頁的最後一行連續印到下一頁，二頁之間沒有一點間隔，這是很惱人的事。假如印表機印到某頁的底端時，能自動空出間隔再跳到下一頁，繼續印，這不是很好嗎？PAGER 將為您做這些工作，它將會計算APPLE 送給印表機所要印的行數。當所要印的行數到達 54 行時，會自動跳 12 行空白行而到下一頁的頂端。PAGER 可以包含在一個程式中使用，或者直接執行。在一頁一頁列印很長的程式時，這是很有用的。

如何使用此程式

PAGER 是寫給順序輸出印表機使用的，它是和APPLE

的順序介面卡連接的。如果平行輸出印表機想使用 PAGER，則此常規中的二個位元組要先改掉。要改變此二位元組，首先載入程式，再打入下列二個指令：

```
POKE 785, 2
POKE 812, 2
```

記着要保存此修改過的程式。

在使用 PAGER 之前，首先要定義每頁所要印的行數，從 BASIC 中打入指令 POKE 798, LP，其中 LP 是每頁所要印的行數。

爲了適應各種不同大小的報表紙，可規定報表紙的長度，從 BASIC 中打入 POKE 804, PL，其中 PL 是每頁長度（英吋）的 6 倍。例如，對一 11 英吋的報表紙，PL = 66。注意，PL 一定要大於 LP。

在啓動 PAGER 之前，若打入指令 POKE 822, 1，則在印表機每印完一頁便會暫停再印。若想要再繼續印，則只要在鍵盤上按下任何鍵即可。當您用手工把報表紙一張一張放在印表機上時，就必須用到這項“暫停輸出”的功能。若不要這暫停動作，則只須打入 POKE 822, 0 指令即可。

包含在一個程式內的指令使用法

在 BASIC 程式中，可利用下列指令將印表機打開與關掉：

```
 5  D$ = CHR$(4)
10  PRINT D$;"PR#1"
20  LW = 132 : REM LINE WIDTH
30  PRINT CHR$(9);LW;"N" : PRINT CHR$(9); "K"
40  CALL 768 : REM TURN ON PAGER
```

（產生輸出結果）

```
50  PRINT D$;"PR#0" : REM TURN PRINTER OFF
```

假如沒有使用DOS，將行號爲 10 的那行改成 PR#1，行號爲 50 的那行改成 PR#0 。如果使用順序印表機，則消除行號爲 20 與 30 二行。

在程式外面的指令使用法

假如使用順序輸出印表機，PAGER 可以由 BASIC的 CALL 768 指令啓動。利用 PR#0 便可解除 PAGER 的啓動狀態。如果使用平行輸出印表機，可執行下列 4 個步驟來啓動 PAGER ：

1. Enter PR#1
2. Enter CTRL-I 132N
3. Enter CTRL-I K
4. Enter CALL 768

可利用指令 PR#0 解除 PAGER 的啓動狀態。

注意事項

1. 在啓動 PAGER 之前，要記得在順序輸出印表機的介面卡上的 DIP 開關上設立適當的資料傳送速度（ baud rate ）。
2. 在啓動 PAGER 之前，要記得調整印表機上的報表紙，以便印出的第一行會印在所要的位置上。
3. 要確定所要印的每行資料寬度不能超過印表機所設的寬度，否則多出的部份資料會印在下一行，而 PAGER 不會把此多印的一行算進去。

```
0800    ;*************************
0800    ;*                       *
0800    ;*    PAGED PRINTER      *
0800    ;*     GARY LITTLE       *
0800    ;*                       *
0800    ;*        PAGE           *
0800    ;*                       *
0800    ;*  COPYRIGHT (C) 1981   *
0800    ;*   MICRO INK, INC.     *
0800    ;* CHELMSFORD, MA 01824  *
0800    ;*  ALL RIGHTS RESERVED  *
0800    ;*                       *
0800    ;*************************
0800    ;
```

```
0800                ;
0800                ;
0800                ;
0800                ;POSITION PAPER IN PRINTER
0800                ;THEN CALL 768 FROM BASIC
0800                ;TO ACTIVATE THIS ROUTINE.
0800                ;TO DE-ACTIVATE, ENT PR#0.
0800                ;
0800                ;PAGE PAUSE FEATURE:
0800                ;  POKE 822,0 TURN OFF
0800                ;  POKE 822,1 TURN ON
0800                ;LINES PRINTED PER PAGE:
0800                ;  PCKE 798,LP
0800                ;
0800                ;PAGE LENGTH:
0800                ;  PCKE 804,PL
0800                ;
0800                ;DESCRIPTICN:
0800                ;  THIS ROUTINE WILL SEND 'PL-LP'
0800                ; BLANK LINES TO THE PRINTER AFTER
0800                ; 'LP' LINES HAVE BEEN SENT BY THE
0800                ; USER.
0800                ;
0800                ;DEFAULTS:
0800                ;  LP=54
0800                ;  PL=66 (11" PAPER)
0800                ;  PAGE PAUSE OFF
0800                ;
0800                ;
0800                ;
0800                ;
0800                COUNT  EPZ $06           ;LINE COUNT STCRAGE
0800                CSWL   EPZ $36           ;OUTPUT HOOK
0800                DOS    EQU $3EA          ;DOS I/O UPDATE HOOK
0800                KBD    EQU $C000         ;KEYBOARD
0800                STRB   EQU $C010         ;KEYBOARD STROBE
0800                PRINT  EQU $C100         ;PR#1 SERIAL OUTPUT
0800                ;
0800                ;
0300                       ORG $300
0300                       OBJ $800
0300                ;
0300                ;
0300 A90F                  LDA #START        ;SET OUTPUT HOCK
0302 8536                  STA CSWL          ;TC START OF RCUTINE.
0304 A903                  LDA /START        ;
0306 8537                  STA CSWL+1
0308 A900                  LDA #$00          ;ZERO THE LINE COUNTER.
030A 8506                  STA COUNT
030C 4CEA03                JMP DOS           ;GIVE NEW HOOK TO DOS.
030F 48             START  PHA               ;RCUTINE STARTS HERE.
0310 2000C1                JSR PRINT         ;SEND CHARACTER TO PRINTER.
0313 68                    PLA
0314 C98D                  CMP #$8D          ;CARRIAGE RETURN?
0316 F001                  BEQ LINE          ;BRANCH IF IT IS.
0318 60             NEXT   RTS
0319 E606           LINE   INC COUNT         ;INCREMENT LINE COUNT.
031B A506                  LDA COUNT
031D C936                  CMP #$36          ;LINE COUNT =54?
031F D0F7                  BNE NEXT          ;IF NOT, THEN RETURN.
```

```
0321 A506     BLANK   LDA COUNT
0323 C942             CMP #$42         ;PAGE LENGTH MET?
0325 F00A             BEQ LOOP
0327 E606             INC COUNT        ;INCREMENT THE COUNTER
0329 A98A             LDA #$8A         ;LOAD A LINE FEED
032B 2000C1           JSR PRINT        ;AND SEND IT TO THE PRINTER
032E 38               SEC
032F B0F0             BCS BLANK
0331 A900     LOOP    LDA #$00         ;ZERO THE COUNTER.
0333 8506             STA COUNT
0335 A900             LDA #$00         ;CHANGE TO LDA #$01 TO
0337 F008             BEQ DONE         ;GET 'PAGE PAUSE'.
0339 2C00C0   AGAIN   BIT KBD          ;WAIT FOR KEYPRESS
033C 10FB             BPL AGAIN        ;BEFORE CONTINUING.
033E 2C10C0           BIT STRB         ;CLEAR KEYBOARD STROBE.
0341 60       DONE    RTS
                      END

***** END OF ASSEMBLY


          **************************
          *                        *
          * SYMBOL TABLE -- V 1.5  *
          *                        *
          **************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

COUNT   0006  CSWL   0036

** ABSOLUTE VARABLES/LABELS

DOS     03EA  KBD    C000  STRB   C010  PRINT  C100
START   030F  NEXT   0318  LINE   0319  BLANK  0321  LOOP   0331  AGAIN  0339
DONE    0341

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:007A
```

十六進位印表機

作者： Leroy Moyer

這個程式允許您對您所要解組合的機器語言指令設立位址的上下界限。

當您使用APPLE II的解組合語言程式（disassembler）以印出機器語言碼時，通常是打入一個起始位址，以及一些‘L’。使用此種方法印出機器語言程式有二個問題。第一，如果機器語言程式的指令個數不是20的倍數，則在印出的最後一個指令之後會緊接着一些亂七八糟的東西印出。第二，當所要印的程式很長時會很麻煩。您想打入50‘L’指令印出所需要的機器語言指令嗎？本程式將解決這二種問題，它可以把在二個指定的十六進位位址之間要解組合的機器語言碼解碼成組合語言指令，並把它印出。

載入此程式之後，使用它便很簡單。首先，利用Control P把印表機打開，然後打入800G（RETURN）。螢幕將會被清除，並打出“STARTING ADDRESS”字樣。接着打入所要解碼的第一個指令的十六進位位址，再按RETURN鍵。同樣的，它也會要求輸入所要解碼的終止

位址，打入終止位址之後，程式將會從起始位址開始重新組回機器語言指令，並且印出來，直到終止位址才停止。

如果要把此程式載入到記憶體的其他位置也是可以的，只要改變二個位址（4個位元組）的值即可。這二個位址內所含的資料就是要求起始位址與終止位址的二行文字。在程式中它們是存在由$86B與$87D開始的位置上，而它們分別是在$806和$828使用。

在此程式中使用很多監督系統的副常規，而我們只說明其中的二個。第一個是GETNUM（$FFA7）副常規，它利用Y暫存器做指標，由緩衝器（$200）中取得以ASCII字元形式儲存的數字，將其轉換成二個位元組的十六進位數字。此副常規將一直轉換ASCII字元，直到它碰到一個不是十六進位數字的字元。RETURN鍵（$8D）在此程式中當作結束字元。轉換後的十六進位位址存在$3E和$3F中，依6502的要求，先存低一位元組，再存高一位元組。

第二個是INSTDSP（$F8D0）副常規。這個副常規將重新組合一個指令，並且顯示在螢幕上。此副程式所要重新組合的運算碼位址是存在PCL（$3A）和PCH（$3B）中。從INSTDSP副常規返回後，將指令的長度減1存在LENGTH（$2F）中。INSTDSP並不改變位址的值（存在$3A、$3B），所以$3A、$3B的值必須加上指令的長度，以便取得下一個運算碼的位址（程式的58行到64行）。

如果您不想在印表機上印出前幾行要求輸入的文字資料

，則只要在程式第55行和56行之間加入一個印表機開啓常規即可。希望這個常規能使您的機器語言輸出形式較爲整齊好看。

```
0800              1  ;**************************
0800              2  ;*                        *
0800              3  ;* HEXIDECIMAL PRINTER    *
0800              4  ;*      LEROY MOYER       *
0800              5  ;*                        *
0800              6  ;*      HEX PRINTER       *
0800              7  ;*                        *
0800              8  ;*  COPYRIGHT (C) 1981    *
0800              9  ;*   MICRO INK, INC.      *
0800             10  ;* CHELMSFORD, MA 01824   *
0800             11  ;*  ALL RIGHTS RESERVED   *
0800             12  ;*                        *
0800             13  ;**************************
0800             14  ;
0800             15  ;DECODE BETWEEN ADR
0800             16  ;
0800             17  FINA    EPZ $FE
0800             18  APA2    EPZ $3E
0800             19  LENG    EPZ $2F
0800             20  APPC    EPZ $3A
0800             21  ;
0800             22          ORG $800
0800             23          OBJ $800
0800             24  ;
0800 2058FC      25  STAR    JSR $FC58    ;CLEAR SCREEN
0803 A200        26          LDX #$00     ;OUTPUT FIRST HEADER LINE
0805 BD6B08      27  DBA2    LDA TIT1,X   ;"STARTING ADDRESS"
0808 F008        28          BEQ DBA1
080A 0980        29          ORA #$80
080C 20EDFD      30          JSR $FDED
080F E8          31          INX
0810 D0F3        32          BNE DBA2
0812 206FFD      33  DBA1    JSR $FD6F    ;KEYBOARD INPUT OF STARTING ADDRESS
0815 A000        34          LDY #$00
0817 20A7FF      35          JSR $FFA7    ;CHANGE TO HEXIDECIMAL ADDRESS
081A A53E        36          LDA APA2     ;MOVE HEXADECIMAL ADDRESS TO
081C 853A        37          STA APPC     ; APPC ($3A)
081E A53F        38          LDA APA2+01
0820 853B        39          STA APPC+01
0822 208EFD      40          JSR $FD8E    ;PRINT LINE FEED
0825 A200        41          LDX #$00     ;PRINT SECOND HEADER LINE
0827 BD7D08      42  DBA4    LDA TIT2,X   ;  "ENDING ADDRESS"
082A F008        43          BEQ DBA3
082C 0980        44          ORA #$80
082E 20EDFD      45          JSR $FDED
0831 E8          46          INX
0832 D0F3        47          BNE DBA4
0834 206FFD      48  DBA3    JSR $FD6F    ;KEYBOARD INPUT OF ENDING ADDRESS
```

```
0837 A000                 LDY #$00
0839 20A7FF               JSR $FFA7       ;CHANGE TO HEXADECIMAL ADDRESS
083C A53E                 LDA APA2        ;MOVE HEXADECIMAL ADDRESS TO
083E 85FE                 STA FINA        ; FINA ($FE) FINAL ADDRESS
0840 A53F                 LDA APA2+01
0842 85FF                 STA FINA+01
0844 208EFD               JSR $FD8E       ;PRINT LINE FEED
0847 20D0F8       DBA5    JSR $F8D0       ;DISASSEMBLE ONE LINE
084A E62F                 INC LENG        ;INCREMENT BYTE FOR LENGTH
084C 18                   CLC
084D A53A                 LDA APPC        ;ADDLENGTH OF INSTRUCTION TO
084F 652F                 ADC LENG        ;ADDRESS THAT IS POINTER FOR
0851 853A                 STA APPC        ;OP CODE TO BE DISASSEMBLED
0853 A53B                 LDA APPC+01
0855 6900                 ADC #$00
0857 853B                 STA APPC+01
0859 38                   SEC
085A A53A                 LDA APPC        ;SUBTRACT FINAL ADDRESS TO SEE IF
085C E5FE                 SBC FINA        ; THE END HAS BEEN REACHED
085E A53B                 LDA APPC+01
0860 E5FF                 SBC FINA+01
0862 90E3                 BCC DBA5
0864 208EFD               JSR $FD8E       ;PRINT LINE FEED
0867 208EFD               JSR $FD8E       ;PRINT LINE FEED
086A 60                   RTS             ;RETURN TO MONITOR
086B D3D4C1       TIT1    ASC "STARTI"    ;DATA FOR FIRST HEADER LINE
086E D2D4C9
0871 CEC7A0               ASC "NG ADD"
0874 C1C4C4
0877 D2C5D3               ASC "RESS "
087A D3A0
087C 00                   HEX 00
087D 0D           TIT2    HEX 0D          ;DATA FOR SECOND HEADER LINE
087E C5CEC4               ASC "ENDING"
0881 C9CEC7
0884 A0C1C4               ASC " ADDRE"
0887 C4D2C5
088A D3D3A0               ASC "SS "
088D 00                   HEX 00
                          END

***** END OF ASSEMBLY


          **************************
          *                        *
          * SYMBOL TABLE -- V 1.5  *
          *                        *
          **************************

LABEL. LOC.   LABEL. LOC.   LABEL. LOC.

** ZERO PAGE VARIABLES:

FINA   00FE   APA2   003E   LENG   002F  APPC 003A
```

```
** ABSOLUTE VARABLES/LABELS

STAR    0800  DBA2   0805
DBA1    0812  DBA4   0827  DBA3   0834  DBA5 0847  TIT1  086B TIT2  087D

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0072
```

3 執行時用的公用程式

簡 介

APPLE II 的共用變數
—— Robert F. Zant

APPLESOFT 的 PRINT USING
—— Gary A. Morris

搜尋字串陣列
—— Gary B. Little

APPLESOFT 與矩陣
—— Cornelis Bongers

AMPER-SORT
—— Alan G. Hill

APPLE II 的追踪列印公共程式
—— Alan G. Hill

第三部份簡介

執行時用的公用程式(Utilities)是一些程式，當其它程式在執行時，它們會提供一些幫助。這種公用程式通常在其它程式執行時鏈結到主程式，而自己本身當作副程式執行。在本章，我們將介紹一些APPLESOFT和整數BASIC程式使用的公用程式，它們將使您的APPLE能力加強。

Robert Zant 的“APPLE II的共用變數”，討論如何在BASIC程式之間設置共用變數位置。計有APPLE-SOFT和整數BASIC二種。Gray Morris的“APPLE-SOFT的PRINT USING”，提供APPLESOFT一個很好用的PRINT指述。Gray Little 的“字串陣列的搜找”，提供一個機器語言的陣列搜尋常規，比相對的BASIC程式要快得多。

下二個公用程式利用到APPLESOFT的ampersand功能，二者都加強APPLESOFT語言的功能。Cornelis Bongers 的“APPLESOFT與矩陣”，利用BASIC陣列提供所有的矩陣運算。Alan Hill的“AMPER-SORT’

，能夠自動將陣列分類，不管是文字或數值陣列。

最後，Alan Hill 的"APPLE II 追踪列印公用程式"，提供交談式（interactive）的追踪整數 BASIC 程式，並可儲存一些追踪資料。

APPLE II 的共用變數

作者：Robert F. Zant

在二個有關聯的程式中傳遞資料要利用共用變數。在此我們利用二個副常規模擬DOS CHAIN的功能，允許在整數 BASIC 或 APPLESOFT BASIC 之下使用共用變數，而不用磁碟。

一些較複雜的問題往往需要寫很多有相互關係的程式來解決。而這些有密切關係的程式往往需要使用許多相同的變數——即共用變數。大部份系統的作法是，當載入新的程式時，不會改變共用變數的值。因此，變數的值可以在某一程式中定義，而留給以後載入的程式使用。

在APPLE II 中並沒有提供處理共用變數的功能。在DOS中的CHAIN命令提供類似的能力，但是它把所有的變數都存起來，而不是只存想要保存的變數。而且只有在DOS 中使用整數 BASIC時才可以用。對於不使用磁碟系統或者APPLESOFT BASIC的程式，APPLE II 並不提供共用變數的功能。

設定公用變數的空間

下列的機器語言常規可以將變數傳給以後載入的程式。整數BASIC和APPLESOFT BASIC 二者均可使用。用法如下：

1. 在執行第一個 BASIC 程式之前要先載入此機器語言常規。
2. 除了最後一個程式以外，在每一個 BASIC程式結束之前或者在執行下一個程式之前要" CALL 774 "。
3. 除了第一個程式以外，在每一個 BASIC程式執行任何會影響變數的值之前要" CALL 770 "。在以後載入的程式中不能再重新定義變數的陣列大小(Dimension)。

因為所有的變數都會存起來，不管需不需要，所以如果在所有的程式都使用相同變數，則主記憶體的使用效率會很好。當然，只有在我們希望所有的程式中都使用共用變數時才要這樣做。其他的記憶體位置則可再宣稱給" 非共用變數"使用。

在APPLESOFT中字串變數將不會正確的保存。假如字串的值是從磁碟、磁帶或鍵盤讀進來，則此值就會被保存起來。假如字串的值是在指定指述(即，A$ = " XXX ")中定義，則此值在以後的程式中將不存在。

處理整數 BASIC 的常規較簡單。只要存取變數表的指標即可。而處理 APPLESOFT 的常規則較麻煩。此常規將非字串變數移到 RAM 較高的位置，即在字串的上面。當下一個程式執行“ CALL 770 ”指令時，此常規就會把這些變數移回到新程式的後面。

```
0800            ;************************
0800            ;*                      *
0800            ;*   COMMON VARIABLES   *
0800            ;*     ROBERT ZANT      *
0800            ;*                      *
0800            ;*      COM-VAR-I       *
0800            ;*                      *
0800            ;*  COPYRIGHT (C) 1981  *
0800            ;*   MICRO INK, INC.    *
0800            ;* CHELMSFORD, MA 01824 *
0800            ;*  ALL RIGHTS RESERVED *
0800            ;*                      *
0800            ;************************
0800            ;
0800            ;
0800            ;FOR INTEGER BASIC
0800            ;
0800            CL       EPZ $1A
0800            CH       EPZ $AB
0800            ;
0800            ;
0302                     ORG $302
0302                     OBJ $800
0302            ;
0302            ;
0302 4C0F03              JMP RECALL          ;ENTRY 770
0305 00                  BRK
0306 A5CC                LDA $CC             ;ENTRY 774 - SAVE VARIABLES
0308 851A                STA CL              ;SAVE END OF
030A A5CD                LDA $CD             ;VARIABLE TABLE
030C 85AB                STA CH
030E 60                  RTS                 ;BACK TO BASIC
030F A51A       RECALL   LDA CL              ;ENTRY 770 - RECALL VARIABLES
0311 85CC                STA $CC             ;RESET END OF
0313 A5AB                LDA CH              ;VARIABLE TABLE
0315 85CD                STA $CD
0317 60                  RTS                 ;BACK TO BASIC
                         END

          **************************
          *                        *
          * SYMBOL TABLE -- V 1.5  *
          *                        *
          **************************

LABEL. LOC.   LABEL. LOC.   LABEL. LOC.
```

```
** ZERO PAGE VARIABLES:

CL      001A   CH      00AB

** ABSOLUTE VARABLES/LABELS

RECALL 030F

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:002A

0800            ;*************************
0800            ;*                       *
0800            ;*   COMMON VARIABLES    *
0800            ;*      ROBERT ZANT      *
0800            ;*                       *
0800            ;*       COM-VAR-A       *
0800            ;*                       *
0800            ;*  COPYRIGHT (C) 1981   *
0800            ;*   MICRO INK, INC.     *
0800            ;* CHELMSFORD, MA 01824  *
0800            ;*  ALL RIGHTS RESERVED  *
0800            ;*                       *
0800            ;*************************
0800            ;
0800            ;
0800            ;
0800            ;FOR APPLESOFT II BASIC
0800            ;
0800            ;
0800            DL      EPZ $18
0800            DH      EPZ $19
0800            CL      EPZ $1A
0800            CH      EPZ $1B
0800            EL      EPZ $1C
0800            EH      EPZ $1D
0800            A1L     EPZ $3C
0800            A1H     EPZ $3D
0800            A2L     EPZ $3E
0800            A2H     EPZ $3F
0800            A4L     EPZ $42
0800            A4H     EPZ $43
0800            ;
0800            ;
0302                    ORG $302
0302                    OBJ $800
0302            ;
0302            ;
0302 4C5603             JMP RECALL    ;ENTRY 770
0305 00                 BRK
0306 38                 SEC           ;ENTRY 774 - SAVE NUMERICS
0307 A56F               LDA $6F       ;COMPUTE ADDRESSES FOR MOVE
0309 8518               STA DL        ; SAVE START OF STRING ADDRESS
030B E56D               SBC $6D       ;END OF NUMERICS
030D 851A               STA CL        ;TEMPORARY STORAGE
030F A570               LDA $70
0311 8519               STA DH
0313 E56E               SBC $6E
0315 851B               STA CH        ;TEMPORARY STORAGE
```

```
0317 18              CLC
0318 A51A            LDA CL
031A 6569            ADC $69          ;START OF NUMERICS
031C 851A            STA CL           ;TEMP STORAGE
031E A51B            LDA CH
0320 656A            ADC $6A
0322 851B            STA CH
0324 A61A            LDX CL           ;SUBTRACT ONE
0326 D002            BNE A1
0328 C61B            DEC CH           ;START OF COMMON
032A CA        A1    DEX
032B 861A            STX CL
032D 8642            STX A4L          ;SET UP MOVE
032F A51B            LDA CH
0331 8543            STA A4H
0333 A569            LDA $69          ;START OF VARIABLES
0335 853C            STA A1L
0337 A56A            LDA $6A
0339 853D            STA A1H
033B A56D            LDA $6D          ;END OF VARIABLES
033D 853E            STA A2L
033F A56E            LDA $6E
0341 853F            STA A2H
0343 A000            LDY #$00
0345 202CFE          JSR $FE2C                 ;USE MONITOR MOVE ROUTINE
0348 38              SEC                       ;COMPUTE DISPLACEMENT
0349 A56B            LDA $6B                   ; TO ARRAYS
034B E569            SBC $69
034D 851C            STA EL
034F A56C            LDA $6C
0351 E56A            SBC $6A
0353 851D            STA EH
0355 60              RTS                       ;BACK TO BASIC
0356 A51A      RECALL LDA CL                   ;ENTRY 770 - RECALL
0358 853C            STA A1L                   ;SET UP MOVE
035A A51B            LDA CH
035C 853D            STA A1H
035E A518            LDA DL                    ;START OF STRINGS
0360 856F            STA $6F
0362 853E            STA A2L
0364 A519            LDA DH
0366 8570            STA $70
0368 853F            STA A2H
036A A569            LDA $69                   ;STYART OF NUMERICS
036C 8542            STA A4L
036E A56A            LDA $6A
0370 8543            STA A4H
0372 A000            LDY #$00
0374 202CFE          JSR $FE2C                 ;USE MONITOR MOVE ROUTINE
0377 18              CLC                       ;COMPUTE START
0378 A569            LDA $69                   ;OF ARRAYS
037A 651C            ADC EL
037C 856B            STA $6B
037E A56A            LDA $6A
0380 651D            ADC EH
0382 856C            STA $6C
0384 38              SEC                       ;COMPUTE END OF NUMERICS
```

```
0385 A56F             LDA $6F
0387 E51A             SBC CL
0389 856D             STA $6D
038B A570             LDA $70
038D E51B             SBC CH
038F 856E             STA $6E          ;TEMP STORAGE
0391 18               CLC
0392 A56D             LDA $6D
0394 6569             ADC $69
0396 856D             STA $6D          ;TEMP VALUE
0398 A56E             LDA $6E
039A 656A             ADC $6A
039C 856E             STA $6E          ;TEMP VALUE
039E A56D             LDA $6D          ;SUBTRACT ONE
03A0 D002             BNE A2
03A2 C66E             DEC $6E          ;END OF NUMERICS
03A4 C66D        A2   DEC $6D
03A6 60               RTS              ;BACK TO BASIC
                     END


          **************************
          *                        *
          * SYMBOL TABLE -- V 1.5  *
          *                        *
          **************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

DL     0018  DH     0019  CL     001A  CH  001B  EL     001C  EH   001D
A1L    003C  A1H    003D  A2L    003E  A2H 003F  A4L    0042  A4H  0043

** ABSOLUTE VARABLES/LABELS

A1     032A  RECALL 0356  A2     03A4

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:008A
```

APPLESOFT的PRINT USING

作者：Gary A. Morris

BASIC 的輸出格式是很惱人的問題。此程式允許使用者在 APPLESOFT 中定義輸出的格式，並且可以很容易的修改。

當我開始使用APPLE 設計商用程式時，最頭痛的問題就是報表的輸出格式。我便開始設計一些 BASIC 副常規來執行這些工作，但是不太實用。試着使用MID$、LEFT$、RIGHT$、與 STR$ ，我決定在 APPLESOFT 中使用 USR 函數來寫一個通用的輸出格式常規。

此常規完全是以組合語言寫的，因爲它在處理此問題時較方便。當您從 BASIC 呼叫此常規時，將一字串指定給變數 ED$ ，此字串就是您想要的輸出格式。當您在 PRINT 敍述中使用 USR 函數時，將會計算參數的值，然後根據 ED$ 變數值的格式印出。

在 BASIC 程式範例（圖 1 ）中，在行號 10 載入此機器語言程式存入RAM中的 $300 ~ $3A9 位置。行號 20 將 " JMP $0300 " 存入 $000A 中，以便 APPLESOFT 能

找到常規的所在。行號 10 和 20 只有在程式開始時才需要用到。行號 30 將編輯符號存入變數 ED$ 。行號 40 是一個使用 USR 函數的簡單 PRINT 敍述。行號 50 將一值指定給X（就是我們想印的），而行號 60 可將X值印出，並只取到小數第二位（四捨五入）。如果想取到小數第三位，則 100 要改成 1000 ，而且在ED$編輯符號中，小數點後要有 3 位數字。注意，在 USR 函數的括號內可以是任何有效的表示式。

此常規將所要印的數字根據編輯符號由右到左印出，其中遇到小數點、逗號、與特殊符號都要跳過。

此常規可以在任何 BASIC 的 PRINT 敍述使用，例如列印到磁碟檔案中，或列印到印表機，或列印到螢幕上。在檔案中建立固定長度記錄（ fixed-length records ）更是有用。

編輯符號可以如圖 1 一樣複雜，或者也可以只是空白。若只有空白符號，將會使得數目字靠右對齊。如果數目字大於編輯符號的長度，則最左邊多出的數字會被切除。數字將會填入編輯符號中空白或數字字元的位置，而跳過一切特殊符號（ $ ， . “ ÷ % ： * ）。

在編輯符號中使用“ 0 ”的作用是，假如數字很小時，小數點和最左邊的數字之間仍會出現“ 0 ”。如果數字太小，小得不會使用到逗號時，則此逗號將會以空白取代。如果在編輯符號中使用小數點，則要列印的數字必須是一個整數，否則會有二個小數點出現。編輯符號的長度不能超過 16

個，如果超過 16 個，多餘的會被切除。

此機器語言程式可以放在程式的任何地方。即不用改變程式而可放在記憶體的任何位置。它需要 169（ $A9 ）個位元組的位置。此程式和 APPLESOFT 佔有相同第 0 頁的位置，所以不會發生衝突。它將編輯符號存在 752 − 767（ $2F0～$2FF ）上。這個地區是在輸入時用的，而在輸出時並不使用它（除了 DOS 的輸出命令）。

它如何工作

從一個 PRINT 敍述開始， BASIC 編譯器便計算USR 函數的參數，然後放在浮點累加器（ accumulator ）。再來利用 JSR 跳到 $000A ，其中有一個 JMP 指令會跳到副常規的起始位置。

在副常規開始執行時，將 APPLESOFT 的浮點累加器轉換成字串（ $300 ～ $30B) ，然後 APPLESOFT 依照所定的格式將其印出。這是由 APPLE 副常規 FPSTR1 與 FPSTR2 來做（我自己命名的）。這些常規將產生的字串放在堆疊（ stack ）（ $100 ）中。

此常規在變數表中尋找ED$（ $30C − $32C ）。找到後，將此值（ $32D − $336 ）移到緩衝器中（ $2F0 − $2FF ）。

當程式擁有所有的資料後，便開始工作。先找到原來數

字的長度（＄337－＄340）；此值（現在是 ASCII 字串）便移到（＄341－＄34D）緩衝器中，然後由右到左一次印出一個字元。編輯符號也要檢查，如果是一個特殊符號便跳過去。負號要在數字的左邊。這個程序將一直執行，直到沒有字元可填或已填滿編輯符號。若未填完，則多餘的逗號以空白取代（＄37A－＄390）。

最後，程式準備將結果印出。＄391－＄39D將會印出所有的數字，除了最後一個數字元（等一下我會說明原因），其中將使用 APPLESOFT 的輸出常規。這個輸出常規將做好所有的檢查與轉換工作，以便 APPLESOFT 的 SPEED、INVERSE、與 FLASH 能正常工作。

USR 函數必須送一個數值給 BASIC 程式，以便讓 BASIC 編譯器將其印出，因為我們是在一個 PRINT 指述中。緩衝器中的最後一個字元（一定是一個數字元）被轉換成整數存在暫存器Y中，並且傳到 APPLESOFT 的整數轉換成實數的常規（＄39E－＄3A8）。這個常規將整數（由暫存器A，Y傳過來的）轉換成浮點值存在浮點累加器中，這就是我們要送回 BASIC 的數值。

硬體方面的要求

此程式需要在有APPLESOFT的APPLE II 或APPLE II PLUS，或是帶有語言卡的APPLE II 上執行。在任

何大小的記憶體中均能執行，不需要使用磁碟驅動機。

在機器語言常規中，如果能適當的修改JSR與JMP，則此程式可和RAM APPLESOFT使用（它載入到$0800－$2FFF）。更改下列各項即可做到此效果：

```
FPSTR1 = $252B
FPSTR2 = $1BDE
COUT   = $135F
INTFP  = $1AEB
FIND   = $184C
```

```
10  REM  PRINT USING DEMO
15  REM
20  POKE 10,76: POKE 11,0: POKE 12,3
30 FD$ = "$   ,   0.00"
40  PRINT "SUB TOTAL..."; USR (3495)
50 X = 12345.67899
60  PRINT "NET TOTAL..."; USR (INT(X*100 +.5))
70  END
```

圖 1

```
0800        ;*************************
0800        ;*                       *
0800        ;*      PRINT USING      *
0800        ;*      GREG MORRIS      *
0800        ;*                       *
0800        ;*  COPYRIGHT (C) 1981   *
0800        ;*   MICRO INK, INC.     *
0800        ;* CHELMSFORD, MA 01824  *
0800        ;*  ALL RIGHTS RESERVED  *
0800        ;*                       *
0800        ;*************************
0800        ;
```

```
0800            ;
0800            ;
0800            ;THE USR FUNCTION REQUIRES A JMP TO
0800            ;THE START OF THE ROUTINE. IF 'START'
0800            ;EQUALS THE ADDRESS WHERE THE ROUTINE
0800            ;IS LOADED THEN THE FOLLOWING WILL SET
0800            ;UP THE JMP:
0800            ;
0800            ; 10 POKE 10,76
0800            ; 20 POKE 11,START-INT(START/256)*256
0800            ; 30 POKE 12,INT(START/256)
0800            ;
0800            ;VARIABLES:
0800            AFLAG  EPZ $52           ;FLAG FOR APPLESOFT
0800            NAME   EPZ $81           ;VARIABLE NAME
0800            PNTR   EPZ $83           ;PNTR TO EDIT PATTERN
0800            VARBLE EPZ $9B           ;POINTER TO VARIABLE
0800            LENGTH EPZ $D0           ;PATTERN LENGTH
0800            ;
0800            ;
0800            ;
0300                   ORG $300          ;ORG AT $0300 (RELOCATABLE)
0300                   OBJ $800
0300            ;
0300            ;
0300            ;
0300            BUFFER EQU $02F0         ;EDIT BUFFER
0300            STRING EQU $0100         ;NUMBER PUT HERE AS
0300            ;  A CHARACTER STRING
0300            ;
0300            ;ROM APPLESOFT SUBROUTINE ADDRESSES:
0300            FPSTR1 EQU $ED34         ;FLOATING TO STRING
0300            FPSTR2 EQU $E3E7         ;CONVERSION ROUTINES
0300            COUT   EQU $DB5C         ;PRINT AN ASCII CHAR
0300            INTFP  EQU $E2F2         ;INT TO FP CONVERSION
0300            FIND   EQU $E053         ;FIND A VARIABLE
0300            ;
0300            ;RAM APPLESOFT SUBROUTINE ADDRESSES:
0300            ;FPSTR1 EQU $252B ;FLOATING TO STRING
0300            ;FPSTR2 EQU $1BDE ;CONVERSION ROUTINES
0300            ;COUT EQU $135F ;PRINT AN ASCII CHAR
0300            ;INTFP EQU $1AEB ;INT TO FP CONVERSION
0300            ;FIND EQU $184C ;FIND A VARIABLE
0300            ;
0300            ;FIRST CONVERT FLOATING POINT ACCUM
0300            ;TO AN ASCII STRING
0300 A552       START  LDA AFLAG         ;SAVE THE FLAG
0302 48                PHA
0303 2034ED            JSR FPSTR1        ;CONVERT FLOATING
0306 20E7E3            JSR FPSTR2        ;POINT TO STRING
0309 68                PLA
030A 8552              STA AFLAG         ;RESTORE FLAG
030C            ;
030C            ;NOW FIND THE VARIABLE (ED$) THAT
030C            ;HAS THE EDIT PATTERN.
030C A945       SEARCH LDA #'E'          ;BASIC VARIABLE
030E A2C4              LDX #$C4          ;NAME IS ED$
0310 8581              STA NAME
0312 8682              STX NAME+1
0314 2053E0            JSR FIND
0317 A004              LDY #4
0319 B19B              LDA (VARBLE),Y    ;GET ADDR HI
```

```
031B 8584          STA PNTR+1
031D 88            DEY
031E B19B          LDA (VARBLE),Y      ;GET ADDR LO
0320 8583          STA PNTR
0322 88            DEY
0323 B19B          LDA (VARBLE),Y      ;GET LENGTH
0325 C910          CMP #16
0327 9002          BCC LENOK           ;MAXIMUM LENGTH
0329 A910          LDA #16             ;ALLOWED IS 16!!!
032B 85DC   LENOK  STA LENGTH
032D        ;MOVE THE PATTERN TO THE BUFFER
032D A8            TAY
032E 88            DEY
032F B183   LOOP2  LDA (PNTR),Y
0331 99F002        STA BUFFER,Y
0334 88            DEY
0335 10F8          BPL LOOP2
0337        ;FIND THE STRING END
0337 A000          LDY #0
0339 B90001 LOOP   LDA STRING,Y        ;GET CHAR
033C F003          BEQ NEXT2
033E C8            INY
033F D0F8          BNE LOOP
0341        ;
0341        ;MOVE STRING TO THE BUFFER, FROM
0341        ;RIGHT TO LEFT, FILLING OVER NUM-
0341        ;BERS BUT SKIPPING COMMA'S AND
0341        ;PERIODS.  IF WE COME TO A MINUS
0341        ;SIGN, THEN KEEP GOING LEFT UNTIL
0341        ;THE PATTERN HAS A BLANK OR A COM-
0341        ;MA, THEN KEEP GOING LEFT STORING
0341        ;BLANKS IN THE BUFFER UNTIL IT ENDS
0341        ;OR WE COME TO A DOLLAR SIGN.
0341        ;
0341 A6DC   NEXT2  LDX LENGTH          ;FIELD WIDTH
0343 88     EDLOOP DEY
0344 B90001        LDA STRING,Y        ;GET A CHARACTER
0347 48            PHA                 ;SAVE IT
0348 68     CHECK  PLA
0349 48            PHA
034A C92D          CMP #'-'            ;IF A MINUS THEN
034C D00E          BNE DIGIT
034E BDEF02 MINUS  LDA BUFFER-1,X
0351 C92D          CMP #'-'
0353 9016          BCC DROPIT
0355 CA     SKIPIT DEX
0356 D0F0          BNE CHECK
0358 68            PLA
0359 18            CLC
035A 9035          BCC DONE
035C BDEF02 DIGIT  LDA BUFFER-1,X
035F C920          CMP #' '
0361 F008          BEQ DROPIT
0363 C93A          CMP #':'
0365 F0EE          BEQ SKIPIT
0367 C930          CMP #'0'
0369 90EA          BCC SKIPIT
036B 68     DROPIT PLA                 ;GET IT BACK
036C 9DEF02        STA BUFFER-1,X
036F CA            DEX
0370 F01F          BEQ DONE
0372 C000          CPY #0              ;END OF STRING?
0374 D0CD          BNE EDLOOP
```

```
0376 E8                   INX
0377 18                   CLC
0378 9010                 BCC NEXT1
037A BDEF02      BLANK    LDA BUFFER-1,X      ;BLANK FROM
037D C924                 CMP #'$'            ;HERE TO $
037F F010                 BEQ DONE
0381 C92E                 CMP #'.'
0383 B005                 BCS NEXT1
0385 A920                 LDA #' '
0387 9DEF02               STA BUFFER-1,X
038A CA          NEXT1    DEX
038B F004                 BEQ DONE
038D E4DC                 CPX LENGTH
038F 90E9                 BCC BLANK
0391 A201        DONE     LDX #1
0393 BDEF02      LOOP4    LDA BUFFER-1,X      ;PRINT THE
0396 205CDB               JSR COUT            ;OUTPUT BUFFER
0399 E8                   INX                 ;EXCEPT LAST CHAR
039A E4DC                 CPX LENGTH
039C 90F5                 BCC LOOP4
039E             ;TAKE THE LAST CHAR FROM THE BUF-
039E             ;FER, CONVERT IT TO FLOATING AND
039E             ;RETURN IT TO APPLESOFT TO BE PRINTED.
039E BDEF02               LDA BUFFER-1,X
03A1 4930                 EOR #'0'
03A3 A8                   TAY                 ;LO ORDER BYTE
03A4 A900                 LDA #0              ;HI ORDER BYTE
03A6 4CF2E2               JMP INTFP           ;CONVERT & RETURN
                          END

***** END OF ASSEMBLY


          **************************
          *                        *
          * SYMBOL TABLE -- V 1.5  *
          *                        *
          **************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

AFLAG  0052  NAME   0081  PNTR   0083  VARBLE 009B  LENGTH 00DC

** ABSOLUTE VARABLES/LABELS

BUFFER 02F0
STRING 0100  FPSTR1 ED34  FPSTR2 E3E7  COUT   DB5C  INTFP  E2F2  FIND   E053
START  0300  SEARCH 030C  LENOK  032B  LOOP2  032F  LOOP   0339  NEXT2  0341
EDLOOP 0343  CHECK  0348  MINUS  034E  SKIPIT 0355  DIGIT  035C  DROPIT 036B
BLANK  037A  NEXT1  038A  DONE   0391  LOOP4  0393

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:00F2
```

搜尋字串陣列

作者： Gary B. Little

這機器語言程式將使得在一個大的字串陣列中的搜尋問題變得更快與更簡單。

您是否曾經想要在某一字串陣列中尋找某一特定字串？您可能會寫一個小的迴路（ loop ）常規來做這工作。然而，假如您必須要做幾萬次的比較才能找到所要的字串，而這可能要花很長的時間。

在此我提供一個很快的尋找常規，如程式所示，它是用 6502 組合語言寫的。

SEARCH常規

要想知道此程式如何工作，首先要分析APPLE在記憶體中儲存變數的方法。詳細情形請看APPLESOFT II BASIC 程式參考手册。對一個一維字串陣列，儲存形式如下：

名稱（ 2 個位元組 ）
位移（ OFFSET ）——指向下一個變數的指標
（ 2 個位元組 ）
維數（ DIMENSION ）（ 1 個位元組 ）
第一維的大小（ 2 個位元組 ）
STRING$（0）——長　度（ 1 個位元組 ）
低位址（ 1 個位元組 ）
高位址（ 1 個位元組 ）
⋮
⋮
STRING$（N）（ 3 個位元組 ）

N是第一維的大小。假如在程式中，字串陣列是第一個定義的陣列變數，則第C個陣列變數的位址是在 PEEK（107）+ 256 * PEEK（108）+ 7 + 3 * C（其中 0 <= C <= N ）。這是因爲指向陣列位址的指標是在 $6B，$6C（107，108），而且在第C個陣列變數之前共有 7 + 3 * C 個位元組。

假如所要尋找的變數是定義在程式中的第一個簡單變數，則字串的位址是在 PEEK（105）+ 256 * PEEK（106）+ 2 ，此變數佔有三個位元組。因爲指向這簡單變數的指標位址是在 $69 ，$64（ 105 ， 106 ）。而在這變數的三個位元組之前有二個位元組。

在尋找時，必須要比較由SV + 3 ，SV + 4 （其中

SV = PEEK (105) + 256 * PEEK (106)) 所指的字串和由AV + 8 + 3 * C , AV + 9 + 3 * C (其中AV = PEEK (107) + 256 * PEEK (108) ; C從0到N) 所指的字串比較。這是在組合語言常規中做的。

執行以下所列的程式便可看出省了不少時間。例如，在 2000 個字串陣列變數中尋找時，只要花 1 秒；而在 APPLESOFT 中要花 19 秒。

使用此搜尋常規

在 APPLESOFT 中使用此常規，其執行程序如下：

1. 將尋找的字串長度，與二個指標分別 POKE 到位址 0 , 6 , 7 。範例程式的行號 210 做此工作。
2. 將正在處理的陣列變數個數 (‘ C ’) POKE 到位址 30 , 31 (低 , 高) 。範例程式的行號 220 做此工作。
3. 將所有陣列變數的個數 (‘ N ’) POKE 到位址 28 , 29 (低 , 高) 。範例程式的行號 230 做此工作。
4. 將第C個陣列變數的位址 POKE 到位址 8 , 9 (低 , 高) 。範例程式的行號 240 做此工作。
5. 利用 CALL 768 開始執行組合語言常規。當控制權送回 APPLESOFT 時，所找到的陣列號碼將存在位址 30 , 31 。假如 PEE (30) + 256 * PEEK (31) 大

於N，則表示找不到。假如找到的話，則放在R$(C)中，其中C = PEEK(30) + 256 * PEEK(31)，R$ 是所尋找的陣列。

6. 繼續對其他的陣列變數尋找，將C加1，重複上述動作。

此常規並不會正確的找到字串陣列變數。如果某一字串陣列變數最左邊部份和所要找的字串一樣，則就當做已經找到了。所以有時會發生錯誤。

```
1   REM  *************************
2   REM  *                       *
3   REM  * STRING SEARCH ROUTINE *
4   REM  *      GARY LITTLE      *
5   REM  *                       *
6   REM  *  COPYRIGHT (C) 1981   *
7   REM  *   MICRO INK, INC.     *
8   REM  * CHELMSFORD, MA 01824  *
9   REM  *  ALL RIGHTS RESERVED  *
10  REM  *                       *
11  REM  *************************
12  REM
14  REM
100 S$ = "": REM  MUST BE FIRST DEFINED SIMPLE VARIABLE
110 N = 2000: DIM R$(N): REM  MUST BE FIRST DEFINED ARRAY VARIABLE
120  GOSUB 1000: REM  LOAD SEARCH ROUTINE
130  DEF  FN MD(X) = X - 256 *  INT (X / 256)
140  TEXT : HOME : PRINT  TAB( 8);: INVERSE : PRINT "STRING ARRAY
     SEARCH  DEMO": NORMAL
150  PRINT : PRINT "RANDOM STRINGS:": PRINT
160  FOR I = 1 TO N:R$(I) =  CHR$ (65 + 26 *  RND (1)) +  CHR$
     (65 + 26 *  RND (1)): PRINT R$(I);" ";: NEXT I: PRINT : PRINT
170  INPUT "ENTER SEARCH STRING: ";S$: PRINT
180 SV = AV:C = 1
190 SV =  PEEK (105) + 256 *  PEEK (106)
200 AV =  PEEK (107) + 256 *  PEEK (108)
210  POKE 0, PEEK (SV + 2): POKE 6, PEEK (SV + 3): POKE 7, PEEK(SV + 4)
220  POKE 30, FN MD(C): POKE 31, INT (C / 256)
230  POKE 28, FN MD(N): POKE 29, INT (N / 256)
240  POKE 8, FN MD(AV + 7 + 3 * C): POKE 9, INT ((AV + 7 + 3 * C) / 256)
250  CALL 768
260 C =  PEEK (30) + 256 *  PEEK (31)
270  IF C > N THEN 300
280  PRINT S$;" MATCHES #";C;" (PHRASE: ";R$(C);")"
290 C = C + 1: IF C <  = N THEN 190
300  PRINT : PRINT "MACHINE LANGUAGE SEARCH COMPLETED"
```

```
310  PRINT : INPUT "PRESS 'RETURN' FOR APPLESOFT SEARCH: ";A$: PRINT
320  FOR I = 1 TO N
330  IF S$ =  LEFT$ (R$(I), LEN (S$)) THEN  PRINT S$;" MATCHES #";I;"
     (PH   RASE: ";R$(I);") "
340  NEXT I: PRINT : PRINT "APPLESOFT SEARCH COMPLETED": END
1000  FOR I = 768 TO 849: READ X: POKE I,X: NEXT I: RETURN
1010  DATA  32,74,255,160,0,177,8,133,1,200,177,8,133,26,200,177,8,133,
      27,165,1,197,0,48,15,160,0,177,6,209
1020  DATA  26,208,7,200,196,0,240,16,208,243,165,30,197,28,208,11,165,
      31,197,29,208,5,230,31,76,63,255,24,165,8
1030  DATA  105,3,144,2,230,9,133,8,24,165,30,105,1,144,2,230,31,133,
      30,56,176,177
0800         ;**************************
0800         ;*                        *
0800         ;*SEARCHING STRING ARRAYS*
0800         ;*     GARY LITTLE        *
0800         ;*                        *
0800         ;*    STRING SEARCH       *
0800         ;*                        *
0800         ;*   COPYRIGHT(C) 1981    *
0800         ;*    MICRO INK, INC.     *
0800         ;*  CHELMSFORD, MA 01824  *
0800         ;*   ALL RIGHTS RESERVED  *
0800         ;*                        *
0800         ;**************************
0800         ;
0800         ;
0800         ;
0800         ;
0800         LENS   EPZ $0       ;LENGTH OF SEARCH PHRASE
0800         LENR   EPZ $1       ;LENGTH OF STRING ARRAY VARIABLE
0800         SP     EPZ $6       ;POINTER TO SEARCH PHRASE
0800         RP     EPZ $8       ;POINTER TO ARRAY VARIABLE TABLE
0800         RL     EPZ $1A      ;POINTER TO ARRAY VARIABLE
0800         NL     EPZ $1C      ;ENDING ARRAY NUMBER
0800         CL     EPZ $1E      ;STARTING ARRAY NUMBER AND COUNTER
0800         SAVE   EQU $FF4A    ;SAVE REGISTERS
0800         RSTORE EQU $FF3F    ;RESTORE REGISTERS
0800         ;
0300                ORG $300
0300                OBJ $800
0300         ;
0300 204AFF         JSR SAVE     ;SAVE REGISTERS
0303 A000    LOOP   LDY #$00
0305 B108           LDA (RP),Y   ;GET LENGTH OF VARIABLE
0307 8501           STA LENR     ;AND STORE
0309 C8             INY
030A B108           LDA (RP),Y   ;GET POINTER (LO)
030C 851A           STA RL       ;AND SAVE
030E C8             INY
030F B108           LDA (RP),Y   ;GET POINTER (HI)
0311 851B           STA RL+1     ;AND SAVE
0313 A501           LDA LENR     ;IF LENGTH OF SEARCH
0315 C500           CMP LENS     ;PHRASE EXCEEDS LENGTH
0317 300F           BMI NOPE     ;OF VARIABLE, SEARCH FAILS
0319 A000           LDY #$00
031B B106    AGAIN  LDA (SP),Y   ;COMPARE THE PHRASES
031D D11A           CMP (RL),Y   ;LETTER BY LETTER
```

```
031F D007                BNE NOPE     ;FAILS IF NOT EQUAL
0321 C8                  INY
0322 C400                CPY LENS
0324 F010                BEQ RTS1     ;SUCCESS!
0326 D0F3                BNE AGAIN
0328 A51E      NOPE      LDA CL       ;COMPARE COUNTER
032A C51C                CMP NL       ;TO ENDING ARRAY NUMBER
032C D00B                BNE LOOP1
032E A51F                LDA CL+1
0330 C51D                CMP NL+1
0332 D005                BNE LOOP1    ;DONE IF EQUAL
0334 E61F                INC CL+1
0336 4C3FFF    RTS1      JMP RSTORE
0339 18        LOOP1     CLC
033A A508                LDA RP       ;SET POINTER TO NEXT
033C 6903                ADC #$03     ;TRIO OF ARRAY BYTES
033E 9002                BCC N1
0340 E609                INC RP+1
0342 8508      N1        STA RP
0344 18                  CLC
0345 A51E                LDA CL       ;INCREMENT COUNTER
0347 6901                ADC #$01
0349 9002                BCC N2
034B E61F                INC CL+1
034D 851E      N2        STA CL
034F 38                  SEC
0350 B0B1                BCS LOOP            ;CHECK NEXT ARRAY VARIABLE
                        END

***** END OF ASSEMBLY


          ***************************
          *                         *
          * SYMBOL TABLE -- V 1.5   *
          *                         *
          ***************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

LENS   0000  LENR   0001  SP     0006  RP     0008  RL     001A  NL  001C
CL     001E

** ABSOLUTE VARABLES/LABELS

SAVE   FF4A  RSTORE FF3F  LOOP   0303  AGAIN  031B  NOPE   0328
RTS1   0336  LOOP1  0339  N1     0342  N2     034D

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0092
```

APPLESOFT 與矩陣

作者：Cornelis Bongers

此機器語言程式能執行大部份的矩陣動作，就如 APPLESOFT 一樣。此程式可利用 & 指述和 APPLESOFT 連絡。使用此程式而不使用 BASIC 程式的優點是，執行速度快和較方便（大概快 5 倍）。在 48K APPLE 其 APPLESOFT 存在 ROM 內（或在語言卡內）的系統才可用此程式。

對於一個矩陣或二個矩陣有許多運算動作。例如，將矩陣A和矩陣B相加，便可得出矩陣C，其元素是矩陣A和矩陣B相對元素的和。所以假如

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 1 & 4 \\ 4 & -2 & 1 \end{bmatrix}$$

且

$$B = \begin{bmatrix} 2 & 4 & 7 \\ 1 & 8 & -6 \\ 5 & 0 & 1 \end{bmatrix}$$

則A與B的和爲

$$C = \begin{bmatrix} 3 & 7 & 12 \\ 3 & 9 & -2 \\ 9 & -2 & 2 \end{bmatrix}$$

很明顯的，矩陣A、B、C可用 BASIC 的二維陣列表示。下列的 BASIC 常規可將矩陣A與矩陣B相加。

```
100 FOR I = 1 TO N: FOR J = 1 TO M: C(I,J) = A(I,J) + B(I,J): NEXTJ,I
```

在我們的例題中，N與M均等於 3 。當使用機器語言程式時，此常規可用下面敍述取代：

```
100 &C = A + B
```

注意，在此敍述中，矩陣的名稱可以任意取。而在 BASIC 常規中，矩陣A、B、C必須配合 BASIC 的陣列名稱。

APPLESOFT的運算動作

除了比較（ comparison ）的運算， SCRN、和CHR$、所有APPLSOFT的運算符號與函數均可在矩陣運算中使用。然而對於矩陣的運算敍述仍有些限制。第一，在一個矩陣運算敍述中所用的矩陣不能超過三個。第二，單值（ single-valued ）表示式（或變數）必須放在括號內。第三，在&指述內所用的矩陣必須宣稱是二維陣列。陣列的大小要大於0，小於255。換句話說，出現在&敍述的矩陣必須先利用 DIM 敍述宣稱。詳細的矩陣運算敍述在“指令節”中會介紹。

例題 1.

```
10  DIM A(10,10): B = 1
20  &A = (B): A = RND(A): A = A*(10): A = INT(A)
```

此例中，將陣列A的所有元素值設爲1。再來對陣列A的所有元素求其 RND 函數值，所以所有A的元素值都介於0與1之間。接着將A乘以 10 ，然後對A的每個元素求其 INT 函數值。執行行號爲 20 的敍述後，A就充滿着0和9之間的隨機數字。注意，A=（RND(1)）將會使A等於相同的隨機數字。

例題 2

```
10 DIM A(5,6), B(5,6), C(5,6)
20 B=3
30 &A=(3): B=(2): C=A*B: C=C∧(B)
```

C = A * B 指述會把A和B相對元素相乘，並把結果存C中相對的元素中。執行完此敍述後，C的所有元素都等於 6 。注意，A、B、C必須宣稱相同大小的陣列。若執行最後一個敍述，則C的元素值會變成 3 次方。假如不是 C = C∧(B) ，而是 C = C∧B ，則C的所有元素均等於 6 的平方，因為現在是矩陣B而不是變數B。

矩陣的運算動作

雖然上述例題所用的運算動作與函數有時很方便，但是很難寫成機器語言程式。此程式眞正的好處不是它能處理 APPLESOFT 的函數與運算動作，而是它能處理一些特殊的運算動作。如下：

1. A = IDN(aexper) 其中A必須是一個方矩陣，且 1 <= aexpr <= N ，N是矩陣A的大小。此敍述將使矩陣A只含有 0 和 1 的值。如果 aexpr = 1 ，則A是一個單位矩陣（ identity matrix ）。如果aexpr

大於 1 ，則單位矩陣的每行（ columns ）會往左旋轉 aexpr － 1行。例如，假如A，B是 3 × 3 的方矩陣，則 A = IDN(1)，B= IDN(2) 會使得

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

2. A = TRN(B) 將令A等於B的轉置矩陣（ transpose ）。假如矩陣B是p × q，則矩陣A一定是q × p。在此並不允許求矩陣本身的轉置矩陣（即 A = TRN(A)）。例如，如果B等於

$$B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 0 \end{bmatrix}$$

則 A = TRN(B) 可得

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 0 \end{bmatrix}$$

3. A＝B・C　將會令A等於B與C的乘積。如果B是p×q，則C一定是q×r，推得A是p×r。等號（'＝'）左邊的矩陣不可以等於等號右邊的任一矩陣。例如，我們可以利用　&C＝A・B　將上述的A、B矩陣相乘，結果爲

$$C=\begin{bmatrix}35 & 14\\ 14 & 20\end{bmatrix}$$

4. A＝MIN（B），A＝MAX（B），或A＝ABM（B）將分別令矩陣A等於矩陣B每行（eolumns）的最小値，或絕對值的最大値。然後把B的所有最小值、最大值、或絕對值最大値存入　A（0，1）。如果B是p×q，則A一定是q×1。

5. A＝INV（B）將會令矩陣A等於矩陣B的反矩陣，並將矩陣B的行列式值（determinant）存入A（0，0）。矩陣A、B必須是大小相同的方矩陣。例如，　D＝INV（C），其中C爲上述的矩陣，可得出

$$D=\begin{bmatrix}.0396825397 & -.0277777778\\ -.0277777778 & .0694444444\end{bmatrix}$$

當執行此敍述時，矩陣D第0列（row）的值會破改掉，因爲它用來存一些指標。如果要取得矩陣A的反矩陣

，可利用 A＝INV（A）。最後，在矩陣的對角線也允許零的存在。

6. A＝NEINV（B）和A＝INV（B） 所得的結果一樣，不同的是當矩陣B是一個單一矩陣（ singular）時，其行列式值爲零，程式仍然會繼續執行。使用NE INV時，在執行完敍述之後會再檢查矩陣B的行列式值（存在A（0，0））。

7. A＝PNT（aexpr） 會顯示矩陣A。對於矩陣A的每個元素，都保留aexpr個位置，並且在顯示每一列之後會自動跳一行。假如aexpr等於零，則矩陣A的每個元素以一個空白分開。

應用例題

一個較有趣的矩陣應用是線性模型（ linear model ）。線性模型能分析一些獨立變數（ independent variables）對從屬變數（ dependent variables ）的影響。形式爲

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots\cdots b_m x_m + u$$

其中y代表從屬變數，x_1、x_2、……等均是獨立變數。

最後一項，u，表示影響程度的因素，並不包括在模型中，例如，假設我們希望建立下列三者之間的關係；第一，

每年度某一特殊產品（ y ）的地域性銷售量；第二，廣告次數（ x_1 ）；第三，在那地域所居住的人口總數（ x_2 ）。資料如下：

Obs. No.	Y Sales	X_1 Advert.	X_2 Popul.
1	118	8	583
2	138	9	692
3	104	5	1082
4	65	1	836
5	46	1	628
6	61	2	244
7	48	1	632
8	66	2	172
9	78	5	319
10	69	2	383

在矩陣代數中此形式可寫成

$$Y = X \cdot B + U$$

其中 B 是 3 × 1（係數未知），Y（銷售量）和 U（餘數）均是 10 × 1。矩陣 X 是 10 × 3。矩陣 X 的第一行（ column ）元素均爲 1（計算 b_0 用），而第二行、第三行中的元素分別等於上表中的 x_1、x_2 行中的元素。爲了配合資料，我們利用最小平方法則（ least squares principle ），就是適當的選擇係數使得矩陣 U 中的元素平方和爲最小。由此可得出矩陣 B 爲

$$B = (X^1 \cdot X)^{-1} X^1 \cdot Y$$

其中 X^1 表示X的轉置矩陣。在下列的 BASIC 程式就是用來計算最小平方值的係數。以上面的例題來說，每增加一些額外的廣告，銷售量便會增加 9.5；若某地域的人口增加 100人，則銷售量會增加 1.6。

在此節我們所提的例題非常簡單。我們可以更擴大的分析大量的資料，也不會困難。這在矩陣代數中只是一小部份而已，但却很有用。其他的應用如 Markor 問題，利用牛頓法（Newton method）對於某些變數求最大值（或最小值）均包括在內。

機器語言程式

此程式從 $8900 開始，計有 700 個位元組。結束於 $8FF2，表示 $9000 － $9600 可以給其它常規使用。（註：此程式並未列出，但是以機器碼的形式存在磁碟中，名稱爲MATRICES。）

利用 BRUN 可以和 APPLESOFT 程式連接，如果您沒有磁碟，則可利用監督系統命令 8900G 。若使用後者，則要利用 Control－C 進入 APPLESOFT。碰到 BRUN 或 8900G 命令時，將會執行一些啓動常規，設立 HIMEM 與 & 向量。如果在執行程式時，不小心把 & 向量毀掉，則可利

用命令CALL 35072重新設立。

此程式儘量使用第0頁記憶體，以便加快處理的速度。然而在執行&指述時 ON ERR 旗號會暫時清除，因爲此矩陣常規使用到 ON ERR 指標的記憶體位置。在執行完&指述後， ON ERR 旗號和指標將會恢復原值。除了第0頁記憶體，使用 Control -Y 和 &向量時，會使得在 $3F5 - $3FA 的值都消失。

當發生錯誤時如何做

假如在執行&敍述時發生錯誤，表示您的程式和我的程式有錯誤。若是您的程式有錯誤，則可能是不符合下列條件:

1. 在&敍述中只允許實數的矩陣。
2. &敍述中所用的矩陣必須是二維的。
3. 矩陣的大小必須大於 0 ，小於 255 。
4. 矩陣安排的次序要符合在“ 指令節 ”所要求的條件。
5. 在&敍述中的矩陣必須先在程式中利用 DIM 指述宣稱。
6. 在執行&敍述時， ON ERR 不會有作用。

而以我的程式來說，比較不可能有錯誤，因爲此程式我已經測試好幾個月了，當然，也許有些錯誤沒有看到，希望發現程式有錯誤的人能告訴我，我將會很感激。

指令節

此節包含執行&敍述時的矩陣表示法。敍述的形式如下：

&矩陣表示式：矩陣表示式：……等等

可以使用下列的運算符號與函數：

運算符號：＝＋、－、＊、／、∧、AND、OR
函　　數：＝SGN、INT、ABS、USR、FRE、PDL、POS、SQR、RND、LOG、EXP、COS、SIN、TAN、ATN、PEEK

在&敍述中的矩陣要注意其大小；在"＝"號左邊的矩陣名稱可以等於"＝"號右邊的矩陣名稱。允許的矩陣表示式如下：

1. 使用APPLESOFT運算符號與函數：

1.1　1個矩陣和1個表示式

A＝（aexpr）

例題：

A＝（－1/2），　B＝（Z%）

1.2　2個矩陣

A = B

A = − B

A = NOT　B

A = function（ B ）

例題：

A = SIN（ B ）

1.3　2個矩陣和1個表示式

A = B　operator（ axepr ）

例題：

A = B ∧（ cos（ − 3 ））

1.4　3個矩陣

A = B　operator　C

例題：

A = B / C

2. 特殊的矩陣運算：

2.1　A = IDN（aexpr）——單位矩陣：A必須是方矩陣，且 1 ≤ aexpr ≤矩陣A的大小。

2.2　A = TRN（B）——轉置矩陣：假如B是 p × q，則A一定是 q × p 。A = TRN（A）是不允許的。

2.3　A = B ・ C——乘法：假如B是 p × q ，C是 q × r ，則A一定是 p × r 。A = A・ C或

A＝C・A是不允許的。

2.4 A＝MIN(B)，A＝MAX(B)，A＝ABM(B)——最小值，最大值，或絕對最大值：假如B是p×q，則A一定是q×1。執行完後，A(0,1)將含着矩陣B中的最小值，最大值或絕對最大值。

2.5 A＝INV(B)——反矩陣：A和B必須是相同大小的方矩陣。執行完後，A(0,0)含着B的行列式值。

2.6 A＝NEINV(B)——反矩陣：和INV相同，不過B若是單一矩陣，程式不會停止執行。

2.7 A＝PNT(aexpr)——列印：假如aexpr＝0，則資料元素以空白分開，否則每個元素保留aexpr個位置。

```
1  REM  *************************
2  REM  *                       *
3  REM  * MATRICES & APPLESOFT *
4  REM  *    BY C. BONGERS      *
5  REM  *                       *
6  REM  *     MATRIX DEMO       *
7  REM  *                       *
8  REM  *  COPYRIGHT (C) 1981   *
9  REM  *   MICRO INK, INC.     *
10  REM * CHELMSFORD, MA 01824 *
11  REM * ALL RIGHTS RESERVED  *
12  REM *                      *
13  REM *************************
14  REM
15  REM  THE LINEAR MODEL
16  REM
18  HOME
20  INPUT "NUMBER OF CBSERVATIONS ? ";N
30  INPUT "NUMBER OF INDEPENDENT VARIABLES ? ";M:M1 = M + 1
40  IF M1 >  = N THEN  PRINT : PRINT "TOO FEW OBSERVATIONS ": STOP
50  DIM X(N,M1),XA(M1,N),Y(N,1),B(M1,1),E(N,1),EA(1,N),S(M1,M1)
```

```
60  DIM V1(1,1),V2(1,1),H(M1,1),J(1,N)
70  PRINT : PRINT "INPUT THE ELEMENTS OF THE Y-VECTOR": PRINT
80  FOR I = 1 TO N
90  PRINT "ELEMENT ";I;" ? ";: INPUT "";Y(I,1):X(I,1) = 1
100  NEXT I
110  FOR J = 2 TO M1
120  PRINT : PRINT "INPUT THE ELEMENTS OF THE X";J - 1;"-VECTOR": PRINT
130  FOR I = 1 TO N
140  PRINT "ELEMENT ";I;" ? ";: INPUT "";X(I,J)
150  NEXT I,J
160  REM  CALCULATE RESULTS
170  & XA = TRN(X):S = XA.X:S = NEINV(S):H = XA.Y:B = S.H
180  IF S(0,0) = 0 THEN  PRINT "THE S-MATRIX IS SINGULAR": STOP
190  PRINT : PRINT "THE LEAST SQUARES EQUATION EQUALS ": PRINT
200  PRINT "Y = ";B(1,1);
210  FOR J = 2 TO M1: IF B(J,1) >  = 0 THEN  PRINT "+";
220  PRINT B(J,1);"*X";J - 1;
230  NEXT : PRINT : PRINT
240  & E = X.B:EA = TRN(E):E = Y - E
250  PRINT "** THE TABLE OF RESIDUALS **": PRINT
260  PRINT "NO"; TAB( 4);"OBSERVED Y"; TAB( 16);"ESTIMATED Y";
     TAB( 29);"   RESIDUAL"
270  FOR I = 1 TO N
280  PRINT I; TAB( 4);Y(I,1); TAB( 16);EA(1,I); TAB( 29);E(I,1)
290  NEXT I: PRINT
300  & EA = TRN(E):V1 = EA.E
310  PRINT "STANDARD DEV. RESIDUALS: "; SQR (V1(1,1) / (N - M1))
320  & J = (1):V2 = J.Y:V2 = V2 / (N):E = Y - (V2(1,1)):EA = TRN(E):
    V2 =  EA.E
330 R = (V2(1,1) - V1(1,1)) / V2(1,1): IF R < 0 THEN R = 0
340  PRINT "R^2";: HTAB (24): PRINT ": "; SQR (R)
350  END
```

AMPER - SORT

作者： Alan G. Hill

這是一個速度很快的機器語言分類（ sort ）共用程式，它能處理APPLE的整數、浮點數字、與文字記錄。因為在 BASIC 能召用此程式，所以此程式是非常有用的。

一個處理記錄（ records ）的應用程式中常需第一個用到的是分類公用程式。如果用 BASIC 來寫此程式，您將會發現它的速度非常慢。而在此我們利用機器語言來寫，將會克服這個困難。處理900個整數、 700 個浮點數字、或300個 30 字元的記錄大概要花費 60 秒。

&連接

AMPER－SORT 不只是好在速度快。它使用到APPLESOFT 的另一特色－－Ampersand ，使BASIC對機器語言的介面更有威力。什麼是Ampersand ，且為什

麼它那麼有用？看看下面的例題，便可知 BASIC 程式如何傳送參數給 AMPER−SORT。

```
100 &SRT#(AB$,0,10,7,10,A,1,5,D)
```

當此指述含在 BASIC 程式中或直接當作命令時，便會命令AMPER−SORT將AB$(0)到AB$(10) 分類，第7到第 10 個字元以漸增次序分類，第1到第5個字元以漸減次序分類。當然，也可利用 POKE 來傳送參數，但一般較喜歡使用 Ampersand 介面。

APPLE的Ampersand 特色並沒有詳細的說明。我是在 1978的十月份CALL APPLE中第一次讀到。當APPLESOFT 編譯器在一個 BASIC 指述的開頭碰到&符號時，它就執行一個JSR $3F5。如果使用者在此放一個JMP 指令，便可和使用者的機器語言常規鏈結（link）。在 CALL APPLE 的十一月份、十二月份期刊中，APPLE也提供一些Ampersand 的處理常規。那些常規使得您的機器語言常規能檢查和轉換 Ampersand 之後的字元或表示式。下列是 AMPER−SORT 所用到的一些常規：

1. CHRGET($00B1)

此常規將會把敍述的下一字元送回到累加暫存器中。

當 JSR $3F5執行時，第一個字元是在累加暫存器中。假如碰到一行的最後字元（00）或敍述結束（$3A）時，

便設立零旗號（ zero flag ）。如果是非數字字元則設立 carry 旗號，若是數字則清除 carry 旗號。在 $B8 和 $B9 的指標會自動增加，所以下一個JSR $B1 才會送回下一個字元。 JSR $B7 會送回一個字元，但不會增加指標值。

2. FRMNUM（$DD67）

此常規將會算出在 Ampersand 敍述中，從現在的指標值到下一個逗點之間的變數表示式與常數。結果存在浮點累加暫存器中。

3. GETADR（$E752）

此常規將會把浮點累加暫存器的值轉換成二個整數值，並存在 $50 和 $51 中。在AMPER-SORT中利用FRM-NUM與GETADR取得一些分類的參數，並轉換成整數。

4. GETBYT（$E6F8）

此常規將取得下一個表示式，並轉換成1個位元組的整數存在X暫存器中。

將$B8和$B9指標指向結束的位置是使用者的責任。

參數說明

參數是以下列形式傳送到 AMPER－SORT：

```
100 &SRT#(AB$,B,E,7,10,A,1,5,D)
```

其中：

AB$　是要分類的字串陣列變數名稱。對字串陣列是 ××$，對整數是××%，對浮點陣列是××。

B　是一個變數、常數、或表示式，表示要分類的陣列從那裏開始，即AB$（B）。

E　是一個變數、常數、或表示式，表示要分類的陣列到那裏停止，即AB$（E）。

7　是一個變數、常數、或表示式，表示所要分類資料主要欄（major field）的起始位置。

10　是一個變數、常數、或表示式，表示所要分類資料主要欄的最後位置。

A　是一個文字，表示主要欄的資料為漸增次序。

1　是一個變數、常數、或表示式，表示所要分類資料次要欄（minor field）的起始位置。

5　是一個變數、常數、或表示式，表示所要分類資料次要欄的最後位置。

D　　是一個文字，表示次要欄的資料爲漸減次序。

如何使用 AMPER-SORT

&SRT　敍述可以對字元、整數、浮點數字分類，而且可直接當作命令使用，或者在 APPLESOFT BASIC 中使用。當然，陣列必須事先定義。

A. 字元陣列

1. 元素長度可相同也可不相同。
2. 全部或一部份元素。
3. 漸增或漸減次序。
4. 可包含一個主要欄和 4 個次要欄。

例題：

```
10 DIM NA$(500)

100 &SRT#(NA$,0,500,1,5,A)
200 &SRT#(NA$,0,500,1,5,A,6,10,D,11,11,A)
299 F% = 0:L = 10
300 &SRT = (NA$,F%,L,10,15,D)
```

行號 100 對全部 501 個元素（ NA$（500））的位置 1 到位置 5 以漸增次序分類。

行號 200 和行號 100 一樣，只不過又定義次要欄。分類時，位置 1 － 5 爲漸增次序，位置 6－10 爲漸減次序，位置 11 爲漸增次序。

行號 299 和 300 對 NA$ (0) 到 NA$ (10) 的位置 10 － 15 ，以漸減次序分類。

B. 整數和浮點數字陣列

1. 全部或部份元素
2. 只允許漸增次序（如果需要漸減次序，則將陣列顛倒）

例題：

```
 10 DIM AB%(100),FP(100)

100 &SRT#(AB%,0,100)
299 S = 50: E = 100
300 &SRT#(AB%,S,E)
399 X = 49
400 &SRT#(FP,0,X)
```

行號 100 對 AB%(100) 的所有 101 個元素以漸增次序分類。行號 299 和 300 對 AB%(50) 到 AB%(100) 分類，行號 399 和 400 對 FP (0) 到 FP (49) 分類。

傳送參數時有一些限制，因此要遵守下列條件：

1. $0 \le B \le E \le$ AB$ 的元素宣稱個數。
2. AB$ 必須是一個數量陣列（ scalar array ）；即 AB$ (10) ，不是 AB$ (20 , 40) 。
3. 要分類的陣列名稱要小於 16 個字元，而只有前二個字元有用，且名稱要唯一。

4. 最多能有 5 個分類欄。
5. 在分類欄的起始位置不能大於最後位置。

下列二項可任意選用：

1. 在陣列註標和分類欄的位置均可使用常數、變數、或表示式。
2. &SRT 命令可直接使用，或在 BASIC 中使用。

在程式中會檢查錯誤

當您看到錯誤消息“? SYNTAX ERROR IN LINE ××× ”您便可知道。假如程式發現沒有變數名稱 AB$，就會發出錯誤消息“VARIABLE ××× NOT FOUND ”。

下面列出 AMPER-SORT 程式，以及一個 BASIC 範例程式。

```
10  REM *************************
20  REM *                       *
30  REM *      AMPER-SORT       *
40  REM *      ALLEN HILL       *
45  REM *                       *
50  REM *    AMPERSORT DEMO     *
55  REM *                       *
60  REM *  COPYRIGHT (C) 1981   *
70  REM *    MICRO INK, INC.    *
80  REM * CHELMSFORD, MA 01824  *
90  REM *  ALL RIGHTS RESERVED  *
100  REM *                       *
110  REM *************************
```

```
1000  GOTO 10000
1050  REM  CHARACTER SORT
1060 CH$ = "ABCDWXYZ":L =  LEN (CH$) - 1
1070 N% = 8
1080  DIM AB$(N%)
1090  FOR I = 0 TO N%
1100 C$ =  MID$ (CH$, INT ( RND (1) * L) + 1,1)
1110 B$ =  MID$ (CH$, INT ( RND (1) * L) + 1,1)
1120  FOR J = 1 TO 3
1130 C$ = C$ + C$:B$ = B$ + B$
1140  NEXT J
1150 AB$(I) = B$ + C$
1160  NEXT I
1170  GOSUB 1240
1180  REM  SORT HALF ASCENDING
1190  REM  SORT HALF DESCENDING
1200  & SRT#(AB$,0,N%,1,8,A,9,16,D)
1210  GOSUB 1260
1220  GOTO 11000
1230  REM  PRINT ROUTINE
1240  PRINT "     BEFORE"
1250  GOTO 1270
1260  PRINT "     AFTER": PRINT "ASCEND  DESCEND"
1270  FOR I = 0 TO N%
1280  PRINT AB$(I): NEXT I: RETURN
2000  REM  INTEGER SORT
2010 N% = 8
2020  DIM IN%(N%)
2030  FOR I = 0 TO N%
2040 IN%(I) = 7500 -  INT ( RND (1) * 15000)
2050  NEXT I
2060  GOSUB 2120
2070  REM    SORT
2080  & SRT#(IN%,0,N%)
2090  GOSUB 2130
2100  GOTO 11000
2110  REM  PRINT ROUTINE
2120  HTAB 10: PRINT "BEFORE": GOTO 2140
2130  HTAB 10: PRINT "AFTER"
2140  FOR I = 0 TO N%
2150  PRINT IN%(I): NEXT I: RETURN
3000  REM  FLOATING POINT
3010 T% = 8
3020  DIM FP(T%)
3030  FOR I = 0 TO 8
3040 FP(I) = 1000 *  RND (1) *  SIN (I * 7.16)
3050  NEXT I
3060  GOSUB 3120
3070  REM  SORT
3080  & SRT#(FP,0,T%)
3090  GOSUB 3130
3100  GOTO 11000
3110  REM  PRINT ROUTINE
3120  HTAB 10: PRINT "BEFORE": GOTO 3140
3130  HTAB 10: PRINT "AFTER"
3140  FOR I = 0 TO T%
3150  PRINT FP(I): NEXT I: RETURN
9999  REM
10000  REM  ** &SORT DEMO **
10010  REM  SAVE ROOM FOR
10020  REM  SORT ROUTINE
10030  HIMEM: 20992: REM  $5200
```

```
10040 D$ =  CHR$ (4)
10050  PRINT D$;"BLOAD AMPERSORT,A$5200"
10060  REM  SET UP '&' HOOK
10070  REM  AT $3F5:JMP $5200
10080  POKE 1013,76: POKE 1014,0: POKE 1015,82
10090  HOME : CLEAR
10100  VTAB 8: HTAB 15: PRINT "SORT DEMO"
10110  PRINT : HTAB 15: PRINT "SELECTIONS"
10120  PRINT : HTAB 10: PRINT "1 INTEGER SORT"
10130  HTAB 10: PRINT "2 FLOATING POINT SORT"
10140  HTAB 10: PRINT "3 CHARACTER SORT"
10150  HTAB 10: PRINT "4 EXIT"
10160  VTAB 17: INPUT "SELECTION ";SE%
10170  IF SE% < 0 OR SE% > 4 THEN 10090
10180  ON SE% GOTO 2000,3000,1050,10190
10190  END
11000  PRINT "HIT ANY KEY TO RETURN TO MENU"
11010  WAIT  - 16384,128
11020  POKE  - 16368,0
11030  GOTO 10090
```

```
0800      ;*************************
0800      ;*                       *
0800      ;*     AMPER-SORT        *
0800      ;*    BY ALAN HILL       *
0800      ;*                       *
0800      ;*      AMPERSORT        *
0800      ;*                       *
0800      ;*  COPYRIGHT (C) 1981   *
0800      ;*   MICRO INK, INC.     *
0800      ;* CHELMSFORD, MA 01824  *
0800      ;*  ALL RIGHTS RESERVED  *
0800      ;*                       *
0800      ;*************************
0800      ;
0800      ;
0800      NAPT   EPZ $D0
0800      NMS1   EPZ $D4
0800      ASI1   EPZ $D6
0800      CSI1   EPZ $D8
0800      ASI2   EPZ $DA
0800      CSI2   EPZ $DC
0800      IIII   EPZ $DE
0800      NNNN   EPZ $E0
0800      FSTR   EPZ $E2
0800      FLEN   EPZ $E7
0800      DISP   EPZ $EC
0800      JJJJ   EPZ $ED
0800      LFNI   EPZ $EF
0800      LENJ   EPZ $F0
0800      TYPE   EPZ $F1
0800      ZZ50   EPZ $50
0800      ZZ6B   EPZ $6B
0800      CHRG   EPZ $B1
0800      GETB   EQU $E6F8       ;APPLESOFT EVALUATION ROUTINE 'GETBYT
0800      SNER   EQU $DEC9       ;OUTPUTS "SYNTAX ERROR"
0800      FRNM   EQU $DD67       ;APPLESOFT EXPRESSION EVALUATOR ROUTI
```

```
0800            GETA    EQU $E752        ;APPLESOFT FP->INT ROUTINE 'GETADR'
0800            MPLY    EQU $558A        ;RELOCATED OLD MON. MULTIPLY ROUTINE
0800            COUT    EQU $FDED        ;APPLESOFT OUTPUT ROUTINE
0800            ;
0800            ;
5200                    ORG $5200
5200                    OBJ $0800
5200            ;
5200            ;
5200            ;PROCESS '&'
5200            ;
5200            ;
5200 48         SORT    PHA                      ;ENTER WITH FIRST CHAR
5201 20E654             JSR SVZP                 ;SAVE A WORK AREA IN ZERO PAGE
5204 68                 PLA
5205 A200               LDX #$00
5207 DD2C55     SR01    CMP SRTS,X               ;EDIT FOR 'SRT#('
520A D046               BNE ERRX                 ;SIGNAL 'SYNTAX ERROR
520C 20B100             JSR CHRG                 ;GET NEXT CHARACTER
520F E8                 INX
5210 E005               CPX #$05
5212 D0F3               BNE SR01
5214 A200               LDX #$00                 ;OK SO FAR
5216 F003               BEQ VNAM
5218 20B100     SR04    JSR CHRG                 ;GET ANOTHER CHARACTER
521B C92C       VNAM    CMP ',                   ;LOOP TO GET ARRAY NAME
521D F00A               BEQ SR05
521F 9D7255             STA NAME,X               ;SAVE NAME
5222 E8                 INX
5223 E010               CPX #$10                 ;16 CHARACTERS IS LONG
5225 D0F1               BNE SR04                 ;ENOUGH FOR A NAME
5227 F029               BEQ ERRX                 ;SIGNAL ERROR
5229 CA         SR05    DEX
522A BD7255             LDA NAME,X               ;WHAT TYPE
522D C924               CMP '$
522F F024               BEQ CHAR                 ;CHARACTER
5231 C925               CMP '%
5233 D015               BNE FP00                 ;FLOATING POINT
5235            ;
5235            ; INTEGER SORT
5235            ;
5235 A201       INTE    LDX #$01                 ;INTEGER
5237 A980       INT1    LDA #$80
5239 1D7255             ORA NAME,X               ;NEG. ASCII
523C 9D7255             STA NAME,X
523F CA                 DEX
5240 10F5               BPL INT1
5242 A902               LDA #$02                 ;INITIALIZE DISPLACEMENT
5244 85EC               STA DISP
5246 A901               LDA #$01
5248 D019               BNE SR06
524A            ;
524A            ;
524A            ;FLOATING POINT SORT
524A            ;
524A            ;
524A A905       FP00    LDA #$05
524C 85EC               STA DISP
524E A902               LDA #$02
5250 D011               BNE SR06
5252            ;
5252            ;
5252 4CA552     ERRX    JMP ERRO
```

```
5255                ;
5255                ;
5255                ;CHARACTER SORT
5255                ;
5255 A980           CHAR    LDA #$80
5257 0D7355                 ORA NAME+01       ;NEG. ASCII
525A 8D7355                 STA NAME+01
525D A903                   LDA #$03
525F 85EC                   STA DISP
5261 A900                   LDA #$00
5263                ;
5263                ; ** SET UP SORT LIMITS **
5263                ;
5263 85F1           SR06    STA TYPE          ;0=CH,1=INT,2=FP
5265 20B100                 JSR CHRG          ;NOW GET SUBSCRIPTS
5268 2067DD                 JSR FRNM          ; AND PUT IN F.P. ACC.
526B 2052E7                 JSR GETA          ;CONVERT TO INTEGER
526E A550                   LDA ZZ50
5270 85DE                   STA IIII          ;FIRST SUBSCRIPT
5272 A551                   LDA ZZ50+01
5274 85DF                   STA IIII+01
5276 20B100                 JSR CHRG
5279 2067DD                 JSR FRNM
527C 2052E7                 JSR GETA
527F A550                   LDA ZZ50
5281 85D4                   STA NMS1          ;LAST SUBSCRIPT INTO N-1
5283 18                     CLC
5284 6901                   ADC #$01
5286 85E0                   STA NNNN          ;N
5288 A551                   LDA ZZ50+01
528A 85D5                   STA NMS1+01
528C 6900                   ADC #$00
528E 85E1                   STA NNNN+01
5290 A5F1                   LDA TYPE
5292 D059                   BNE TERM          ;BRANCH NOT CHARACTER SORT
5294 F015                   BEQ SR16
5296                ;
5296                ; **** ERROR ****
5296                ;
5296 A200           ERR3    LDX #$00
5298 BD3155         SR11    LDA MSG1,X        ;ARRAY VARIABLE NAME
529B 0980                   ORA #$80          ;NOT FOUND
529D 20EDFD                 JSR COUT          ;NOTIFY USER
52A0 E8                     INX
52A1 E017                   CPX #$17
52A3 D0F3                   BNE SR11
52A5 200955         ERRO    JSR RSZP          ;RESTORE ZERO PAGE AND
52A8 4CC9DE                 JMP SNER          ;SIGNAL SYNTAX ERROR
52AB                ;
52AB                ; *** GET SORT FIELDS ***
52AB A000           SR16    LDY #$00
52AD 8C8955                 STY SAVY
52B0 20B100         SR17    JSR CHRG          ;GET NEXT CHARACTER
52B3 20F8E6                 JSR GETB
52B6 CA                     DEX
52B7 AC8955                 LDY SAVY
52BA 96E2                   STX FSTR,Y        ;START COLUMN-1
52BC 20B100                 JSR CHRG
52BF 20F8E6                 JSR GETB
52C2 AC8955                 LDY SAVY
52C5 96E7                   STX FLEN,Y        ;END COLUMN
52C7 20B100                 JSR CHRG
52CA 90D9                   BCC ERRO          ;SHOULD BE 'A'OR'D'
52CC C944                   CMP 'D
```

```
52CE F004            BEQ SR07        ;DESCENDING
52D0 A9FF            LDA #$FF        ;ASCENDING
52D2 3002            BMI SR09
52D4 A900    SR07    LDA #$00
52D6 998255  SR09    STA UPDN,Y      ;SAVE SEQUENCE
52D9 C8              INY
52DA 8C8955          STY SAVY
52DD 20B100          JSR CHRG
52E0 C929            CMP ')
52E2 F006            BEQ LAST
52E4 C92C            CMP ',
52E6 F0C8            BEQ SR17        ;LOOP FOR NEXT SORT FIELDPARMS
52E8 D0BB            BNE ERRO
52EA 8C8855  LAST    STY PRSN        ;NO. OF SORT FIELDS
52ED 20B100  TERM    JSR CHRG        ;MUST BE TERMINATOR
52F0 D0B3            BNE ERRO        ;IT WASN'T
52F2         ;
52F2         ;SEARCH SORT ARRAY NAME
52F2         ;
52F2 A000    MC20    LDY #$00
52F4 B16B            LDA (ZZ6B),Y
52F6 CD7255          CMP NAME
52F9 D008            BNE MC22
52FB C8              INY             ;FOUND FIRST CHARACTER
52FC B16B            LDA (ZZ6B),Y
52FE CD7355          CMP NAME+01
5301 F02B            BEQ SETN        ;FOUND BOTH
5303 18      MC22    CLC             ;KEEP LOOKING
5304 A002            LDY #$02
5306 B16B            LDA (ZZ6B),Y
5308 656B            ADC ZZ6B
530A 48              PHA
530B C8              INY
530C B16B            LDA (ZZ6B),Y
530E 656C            ADC ZZ6B+01
5310 856C            STA ZZ6B+01
5312 68              PLA
5313 856B            STA ZZ6B
5315 C56D            CMP $6D
5317 A56C            LDA ZZ6B+01
5319 E56E            SBC $6E
531B B003            BCS SR27        ;NO LUCK, OUT OF BOUNDS
531D 4CF252          JMP MC20
5320         ;
5320         ; ****** NAME NOT FOUND *******
5320         ;
5320 A202    SR27    LDX #$02
5322 BD7255  SR28    LDA NAME,X
5325 9D3B55          STA VARI+1,X    ;PUT NAME IN BUFFER
5328 CA              DEX
5329 10F7            BPL SR28
532B 4C9652          JMP ERR3        ;SEND A MESSAGE
532E         ;
532E         ; ***** INITIALIZE ARRAY POINTER ***
532E         ;
532E 18      SETN    CLC             ;FOUND VARIABLE NAME OF
532F A56B            LDA ZZ6B        ;ARRAY TO BE SORTED.
5331 6907            ADC #$07        ;COMPUTE ADDRESS OF
5333 8552            STA $52         ;STRING LENGTH BYTE.
5335 A56C            LDA ZZ6B+01
5337 6900            ADC #$00
5339 8553            STA $53
533B A5DE            LDA IIII        ;(6B,6C)+7+DISP*IIII
```

```
533D 8550           STA ZZ50
533F A5DF           LDA IIII+01
5341 8551           STA ZZ50+01
5343 A5EC           LDA DISP
5345 8554           STA $54
5347 A900           LDA #$00
5349 8555           STA $55
534B 208A55         JSR MPLY          ;ROM MULTIPLY ROUTINE
534E A550           LDA ZZ50
5350 85D6           STA ASII          ;SAVE ADDRESS FOR MUCH USE
5352 A551           LDA ZZ50+01
5354 85D7           STA ASII+01
5356 4C6653         JMP SR22
5359               ;
5359               ;****** BEGIN SORT *******
5359               ;
5359               ;** FOR I =II TO N-1 LOOP **
5359               ;
5359 18            CONI  CLC
535A A5D6           LDA ASII
535C 65EC           ADC DISP          ;NEXT I ADDRESS
535E 85D6           STA ASII
5360 A5D7           LDA ASII+01
5362 6900           ADC #$00
5364 85D7           STA ASII+01
5366 A001          SR22  LDY #$01
5368 B1D6           LDA (ASII),Y      ;GET ADDRESS OF THE
536A 85D8           STA CSII          ;CHARACTER STRING
536C C8             INY
536D B1D6           LDA (ASII),Y
536F 85D9           STA CSII+01
5371 18             CLC
5372 A5D6           LDA ASII          ; ALSO NEED ADDRESS OF
5374 65EC           ADC DISP          ;ADJACENT ELEMENT FOR
5376 85DA           STA ASI2          ;BUBBLE SORT COMPARISON
5378 A5D7           LDA ASII+01
537A 6900           ADC #$00
537C 85DB           STA ASI2+01
537E 18             CLC
537F A5DE           LDA IIII
5381 6901           ADC #$01
5383 85ED           STA JJJJ          ;J=I+1
5385 A5DF           LDA IIII+01
5387 6900           ADC #$00
5389 85EE           STA JJJJ+01
538B 4C9B53         JMP SR24
538E               ;
538E               ;**** FOR J=I+1 TO N LOOP ****
538E               ;
538E 18            CONJ  CLC
538F A5DA           LDA ASI2
5391 65EC           ADC DISP          ;INCREMENT ABS(J) ADDRESS
5393 85DA           STA ASI2
5395 A5DB           LDA ASI2+01
5397 6900           ADC #$00
5399 85DB           STA ASI2+01
539B A001          SR24  LDY #$01
539D B1DA           LDA (ASI2),Y
539F 85DC           STA CSI2          ;GET NEW STRING ADDRESS
53A1 C8             INY
53A2 B1DA           LDA (ASI2),Y
53A4 85DD           STA CSI2+01
53A6 A5F1           LDA TYPE
```

```
53A8 F003          BEQ CHST          ;CHARACTER SORT
53AA 4C2F54        JMP NCHH
53AD         ;
53AD         ;*** CHARACTER SORT ***
53AD         ;
53AD A000    CHST  LDY #$00
53AF B1D6          LDA (ASI1),Y      ;STRING LENGTH
53B1 F052          BEQ MC40          ;NULL STRING: SKIP
53B3 85EF          STA LENI          ;SAVE LEN (AB$(I))
53B5 B1DA          LDA (ASI2),Y
53B7 F04C          BEQ MC40
53B9 85F0          STA LENJ          ;SAVE LEN(AB$(J))
53BB A200          LDX #$00
53BD B4E2    SR29  LDY FSTR,X        ;STARTING SORT COLUMN
53BF BD8255  MC33  LDA UPDN,X        ;SEQUENCE
53C2 300C          BMI ASND          ;BRANCH ASCENDING
53C4 B1D8          LDA (CSI1),Y      ;CHARACTER BY CHARACTER
53C6 D1DC          CMP (CSI2),Y      ;COMPARISON FOR DESCENDING
53C8 B014          BGE MC26          ;POSSIBLE SWAP
53CA 20C154        JSR SWAP          ;DEFINITE SWAP
53CD 4C0554        JMP MC40          ;NEXT RECORD
53D0 B1D8    ASND  LDA (CSI1),Y      ;ASCENDING
53D2 D1DC          CMP (CSI2),Y
53D4 902F          BLT MC40          ;NO SWAP: NEXT RECORD
53D6 F019          BEQ MC27          ;POSSIBLE SWAP
53D8 20C154  MC25  JSR SWAP          ;SWAP
53DB 4C0554        JMP MC40          ;NEXT RECORD
53DE D025    MC26  BNE MC40          ;NO SWAP
53E0 C8            INY               ;LOOK AT REMAINING CHARACTER
53E1 C4EF          CPY LENI
53E3 F006          BEQ MC39          ;UP TO THE LIMITS OF UNTIL
53E5 C4F0          CPY LENJ
53E7 F016          BEQ MC29          ;WE FIND A REASON TO SWAP
53E9 900F          BLT MC28
53EB C4F0    MC39  CPY LENJ
53ED 90E9          BLT MC25          ;SWAP
53EF F00E          BEQ MC29          ;NO SWAP
53F1 C8      MC27  INY
53F2 C4FF          CPY LENI
53F4 F009          BEQ MC29
53F6 C4F0          CPY LENJ
53F8 F0DE          BEQ MC25
53FA 98      MC28  TYA
53FB D5E7          CMP FLEN,X        ;END OF SORT FIELD?
53FD D0C0          BNE MC33          ;BRANCH NO
53FF E8      MC29  INX
5400 EC8855        CPX PRSN          ;YES, ANY MORE FIELDS?
5403 D0B8          BNE SR29
5405         ;
5405         ;***** NEXT J ********
5405         ;
5405 E6ED    MC40  INC JJJJ
5407 D002          BNE MC38
5409 E6EE          INC JJJJ+01       ;J=J+1
540B A5ED    MC38  LDA JJJJ
540D C5E0          CMP NNNN          ;J=N?
540F A5EE          LDA JJJJ+01
5411 E5E1          SBC NNNN+01
5413 9014          BCC JMPJ          ;BRANCH NO
5415         ;
5415         ;*** NEXT I ****
5415         ;
5415 E6DE          INC IIII
```

```
5417 D002            BNE MC41
5419 E6DF            INC IIII+01         ;I=I+1
541B A5DE      MC41  LDA IIII
541D C5D4            CMP NMS1            ;I=N-1?
541F A5DF            LDA IIII+01
5421 E5D5            SBC NMS1+01
5423 9007            BCC JMPI            ;BRANCH NO
5425           ;
5425           ;***** SORT DONE *******
5425           ;
5425 200955    SDON  JSR RSZP            ;RESTORE ZERO PAGE
5428 60              RTS
5429 4C8E53    JMPJ  JMP CONJ
542C 4C5953    JMPI  JMP CONI
542F 18        NCHH  CLC                 ;NOT A CHARACTER SORT.SO
5430 6A              ROR                 ;IT MUST BE INTEGER OR F. P
5431 B003            BCS INTC            ;IT'S INTEGER
5433 4C6D54          JMP FPCC            ;IT'S FLOATING POINT
5436           ;
5436           ;***** INTEGER SORT ******
5436           ;
5436 A001      INTC  LDY #$01
5438 B1D6            LDA (ASI1),Y        ;ASCENDING ORDER ONLY
543A D1DA            CMP (ASI2),Y
543C 88              DEY                 ;COMPARE IN%(I) WITH IN%(J)
543D B1D6            LDA (ASI1),Y
543F F1DA            SBC (ASI2),Y
5441 9022            BCC NCSP            ;POSSIBLE SWAP
5443 B1D6            LDA (ASI1),Y
5445 51DA            EOR (ASI2),Y
5447 30BC            BMI MC40
5449           ;
5449           ;**** SWAP I WITH J *******
5449           ;
5449 C8        SWIN  INY
544A B1DA            LDA (ASI2),Y
544C 48              PHA
544D 88              DEY
544E B1DA            LDA (ASI2),Y        ;SWAP IN%(I) WITH IN%(J)
5450 48              PHA
5451 B1D6            LDA (ASI1),Y
5453 91DA            STA (ASI2),Y
5455 C8              INY
5456 B1D6            LDA (ASI1),Y
5458 91DA            STA (ASI2),Y
545A 88              DEY
545B 68              PLA
545C 91D6            STA (ASI1),Y
545E C8              INY
545F 68              PLA
5460 91D6            STA (ASI1),Y
5462 4C0554          JMP MC40            ;NEXT RECORD
5465 B1D6      NCSP  LDA (ASI1),Y
5467 51DA            EOR (ASI2),Y
5469 30DE            BMI SWIN            ;SWAP
546B 1098            BPL MC40
546D           ;
546D           ; **** FLOATING POINT SORT ****
546D           ;
546D A000      FPCC  LDY #$00
546F B1D6      FP01  LDA (ASI1),Y
5471 D1DA            CMP (ASI2),Y
5473 900B            BCC MBSP
```

```
5475 F002          BEQ FPQ2
5477 B01D          BCS FPSP          ;THIS BIT OF CONVOLUTED
5479 C8      FPQ2  INY               ;LOGIC TELLS ME IF
547A C005          CPY #$05          ;FP(I) IS GREATER THAN,
547C D0F1          BNE FPQ1          ;EQUAL TO, OR LESS THAN
547E F03E          BEQ JM40          ;FP(J).
5480 A001    MBSP  LDY #$01
5482 B1D6          LDA (ASI1),Y      ;A TRUTH TABLE HELPS
5484 31DA          AND (ASI2),Y
5486 11DA          ORA (ASI2),Y
5488 3020          BMI FPQ3
548A 88            DEY
548B B1DA          LDA (ASI2),Y
548D D02F          BNE JM40
548F C8            INY
5490 B1D6          LDA (ASI1),Y
5492 1016          BPL FPQ3
5494 3028          BMI JM40
5496 A001    FPSP  LDY #$01
5498 B1D6          LDA (ASI1),Y
549A 31DA          AND (ASI2),Y
549C 11D6          ORA (ASI1),Y
549E 301E          BMI JM40
54A0 88            DEY
54A1 B1D6          LDA (ASI1),Y
54A3 D0C5          BNE FPQ3
54A5 C8            INY
54A6 B1DA          LDA (ASI2),Y
54A8 1014          BPL JM40
54AA A004    FPQ3  LDY #$04
54AC B1D6    FPQ4  LDA (ASI1),Y      ;SAVE FP(I) IN STACK
54AE 48            PHA
54AF 88            DEY
54B0 10FA          BPL FPQ4
54B2 C8      FPQ8  INY
54B3 B1DA          LDA (ASI2),Y
54B5 91D6          STA (ASI1),Y      ;SWAP
54B7 68            PLA
54B8 91DA          STA (ASI2),Y
54BA C004          CPY #$04
54BC D0F4          BNE FPQ8
54BE 4C0554  JM40  JMP MC40          ;NEXT RECORD
54C1 A000    SWAP  LDY #$00
54C3 B1D6          LDA (ASI1),Y
54C5 48            PHA               ;ROUTINE TO SWAP THE
54C6 C8            INY
54C7 A5D8          LDA CSI1          ;CHARACTER POINTERS FOR
54C9 91DA          STA (ASI2),Y
54CB C8            INY               ;CHARACTER SORT.
54CC A5D9          LDA CSI1+01
54CE 91DA          STA (ASI2),Y
54D0 A5DD          LDA CSI2+01
54D2 91D6          STA (ASI1),Y
54D4 85D9          STA CSI1+01
54D6 88            DEY
54D7 A5DC          LDA CSI2
54D9 91D6          STA (ASI1),Y
54DB 85D8          STA CSI1
54DD 88            DEY
54DE B1DA          LDA (ASI2),Y
54E0 91D6          STA (ASI1),Y
54E2 68            PLA
54E3 91DA          STA (ASI2),Y
```

```
54E5 60                  RTS
54E6 A200        SVZP    LDX #$00           ;SAVE SOME OF APPLESOFT'S
54E8 B5D0        MC51    LDA NAPT,X         ;ZERO PAGE. SORT ROUTINE
54EA 9D4855              STA ZPSV,X         ;NEEDS SOME ROOM TO WORK.
54ED E8                  INX
54EE E022                CPX 22
54F0 D0F6                BNE MC51
54F2 A56B                LDA ZZ6B           ;ALSO $6B.6C
54F4 8D7055              STA SV6B
54F7 A56C                LDA ZZ6B+01
54F9 8D7155              STA SV6B+01
54FC A200                LDX #$00
54FE B550        MC55    LDA ZZ50,X         ;ALSO $50.55
5500 9D6A55              STA SV50,X
5503 E8                  INX
5504 E006                CPX #$06
5506 D0F6                BNE MC55
5508 60                  RTS
5509 A200        RSZP    LDX #$00           ;RESTORE ZERO PAGE DATA
550B BD4855      MC61    LDA ZPSV,X
550E 95D0                STA NAPT,X
5510 E8                  INX
5511 EC22                CPX 22
5513 D0F6                BNE MC61
5515 AD7055              LDA SV6B
5518 856B                STA ZZ6B
551A AD7155              LDA SV6B+01
551D 856C                STA ZZ6B+01
551F A200                LDX #$00
5521 BD6A55      MC65    LDA SV50,X
5524 9550                STA ZZ50,X
5526 E8                  INX
5527 E006                CPX #$06
5529 D0F6                BNE MC65
552B 60                  RTS
552C             ;
552C 535254      SRTS    ASC 'SRT#('
552F 2328
5531 8D          MSG1    HEX 8D
5532 564152              ASC 'VARIABLE'
5535 494142
5538 4C45
553A 202020      VARI    HEX 2020202020
553D 2020
553F 4E4F54              ASC 'NOT FOUND'
5542 20464F
5545 554E44
5548 000000      ZPSV    HEX 0000000000000000
554B 000000
554E 0000
5550 000000              HEX 0000000000000000
5553 000000
5556 0000
5558 000000              HEX 0000000000000000
555B 000000
555E 0000
5560 000000              HEX 0000000000000000
5563 000000
5566 0000
5568 0000                HEX 0000
556A 000000      SV50    HEX 000000000000
556D 000000
5570 0000        SV6B    HEX 0000
```

```
5572 000000   527   NAME     HEX 0000000000000000
5575 000000
5578 0000
557A 000000   528            HEX 0000000000000000
557D 000000
5580 0000
5582 000000   529   UPDN     HEX 0000000000
5585 0000
5587 00       530   INDS     HEX 00
5588 00       531   PRSN     HEX 00
5589 00       532   SAVY     HEX 00
             533           END

***** END OF ASSEMBLY
```

```
          *************************
          *                       *
          * SYMBOL TABLE -- V 1.5 *
          *                       *
          *************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

NAPT   00D0  NMS1   00D4  ASII   00D6  CSII   00D8  ASI2   00DA  CSI2   00DC
IIII   00DE  NNNN   00E0  FSTR   00E2  FLEN   00E7  DISP   00EC  JJJJ   00ED
LENI   00EF  LENJ   00FC  TYPE   00F1  ZZ50   0050  ZZ6B   006B  CHRG   00B1


** ABSOLUTE VARABLES/LABELS

GETB   E6F8  SNER   DEC9  FRNM   DD67  GETA   E752  MPLY   558A  COUT   FDED
SORT   5200  SR01   5207  SR04   5218  VNAM   521B  SR05   5229  INTE   5235
INT1   5237  FP00   524A  ERRX   5252  CHAR   5255  SR06   5263  ERR3   5296
SR11   5298  ERRC   52A5  SR16   52AB  SR17   52B0  SR07   52D4  SR09   52D6
LAST   52EA  TERM   52ED  MC20   52F2  MC22   5303  SR27   5320  SR28   5322
SETN   532E  CONI   5359  SR22   5366  CONJ   538E  SR24   539B  CHST   53AD
SR29   53BD  MC33   53BF  ASND   53D0  MC25   53D8  MC26   53DE  MC39   53EB
MC27   53F1  MC28   53FA  MC29   53FF  MC40   5405  MC38   540B  MC41   541B
SDON   5425  JMPJ   5429  JMPI   542C  NCHH   542F  INTC   5436  SWIN   5449
NOSP   5465  FPCC   546D  FP01   546F  FP02   5479  MBSP   5480  FPSP   5496
FP03   54AA  FP04   54AC  FP08   54B2  JM40   54BE  SWAP   54C1  SVZP   54E6
MC51   54E8  MC55   54FE  RSZP   5509  MC61   550B  MC65   5521  SRTS   552C
MSG1   5531  VARI   553A  ZPSV   5548  SV50   556A  SV6B   5570  NAME   5572
UPDN   5582  INDS   5587  PRSN   5588  SAVY   5589

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0332
```

APPLE II 的追踪列印公用程式

作者： **Alan G. Hill**

整數 BASIC 的追踪命令對於程式的除錯提供許多有用的資料。但是這些資料的格式並不怎麼好用也不好看（一連串的行號）。此公用程式將提供較易明瞭的輸出格式，並且對較長的追踪保留行號。

您曾經在整數 BASIC 中使用 TRACE 函數，而因爲整個螢幕充滿了行號而絕望嗎？如果使用它而不用印表機，則您可能不會再去使用 TRACE 了！此公用程式會把 TRACE 放回您的除錯工具中。（爲了有些人要使用它）

此公用程式將會列印每一個按次序執行的 BASIC 敍述。不須要在 TRACE 的行號和列印的原始程式中往前看或往後看。在此有二種型式：第一種是即時公用程式 (real-time utility)；即每執行一個敍述便列印一個敍述，所以您可以跟隨程式的流程，您可以減慢執行的速度或暫時停止執行。第二種型式只保留最後執行的 100 行敍述的行號，而最後執行完才列印。在追踪一個全螢幕（ full-screen ）圖形程

式時，第二種型式很有用。

應用技巧

此程式利用DOS COUT連接於$AA53、$AA54，以便截斷和抑制TRACE結果的輸出。其它的輸出都正常，除了一點例外（看警告＃1）。在回到BASIC之前，取得行號並存在LOMEM之上的陣列（TR）中。如果行號和前一個行號一樣，便在堆疊中放一個行號0的指述，例如一個遲延廻路 FOR I = 1 TO 1000 : NEXT I 。當行號改變，就會放入堆疊中。最近的100個行號將會被存起來。使用者可利用TRACE和NOTRACE指述來控制是否要追踪程式。在第二種型式中，在執行完程式時才會列印出行號。第一種型式也是類似作法，只有一點不同。截斷TRACE的常規（TRACE INTERCEPT ROUTINE）將控制權傳給此公用程式，只要有行號存入堆疊中，便將其列印出來。

截斷TRACE的常規如何工作

在$AA53、$AA54的輸出指標由此公用程式設爲截斷TRACE常規（TIR）的位址（$300）。假如TRACE旗號（$AO的第7個位元）爲開（ON），則TIR將檢查

每一字元。如果是關（OFF），TIR將跳回正常的列印公用程式於$FDF0。假如字元是一個＃（$A3），就假設跟隨着一個行號。每一個行號將壓入堆疊中，所以最新的行號一定在堆疊的頂端。$DC、$DD指到下一個要執行的BASIC敍述。行號是在第2和第3個位元組中。在第2種型式，TIR將返回編譯器。而在即時系統（第1種型式）,控制權將傳送到公用程式的行號30020。TIR期望此公用程式30010的CALL 945將行號30010的位址存在$15、$16中。TIR首先保存$DC、$DD，然後再取代$15、$16的值。它也保存現時的BASIC敍述的位址。即$E0、$E1所含的值存在$1B、$1C。TIR便可利用JMP $E88A將控制權傳回編譯器，JMP $E88A會執行公用程式的行號30020。測試程式的現時行號會列印出來；BASIC的指標由行號30090的CALL 954恢復；彈出回應位址；控制權經由$E881返回測試程式。

如前面所提，TR陣列是用來存行號。當進入TIR時便要將陣列設好。注意，TR陣列在公用程式中並未用DIM敍述宣稱。TIR必須要做此工作，因爲對測試程式RUN一次會使變數地區指標（$CC，$CD）重設爲 LOMEM 。

設計此常規

TIR由$300開始。如果改變POKE和CALL 敍述的

絕對參考值，便可放在記憶體的任何位置。同時也要注意在 30060 和 32040 的 LIST 敍述在語句上是不能接受的。所以必須先換成 PRINT 敍述，定位址，再利用監督系統改變成 LIST 符號（$74）。如果在打入其它行之前，先把這些行改好，那麼就比較簡單了。看看下列的例題，其中 HIMEM 是 32768：

```
NEW
30060 PRINT EXECLINE
32040 PRINT TR (I)
CALL  - 151 (to enter Monitor)
*7FEC:74
*7FF9:74
(enter Control/C)
LIST
30060 LIST EXECLINE
32040 LIST TR (I)
```

公用程式的使用

1. 所要測試的程式要按在公用程式之後。
2. 產生一行號 0 的敍述，用以表示某行已正確的執行。例如：

   0 REM *** ABOVE LINE REPEATED ***

3. 可自由選擇執行公用程式：

   ① RUN 30000 ——第一種型式（即時列印）

② RUN 32000 ——第二種型式（執行完才印列）

4. 在測試程式中視其需要可加入 TRACE/NOTRACE 敍述。如果想追踪整個程式就直接打入TRACE 命令。請看警告# 1。

5. RUN 測試程式。

6. 顯示結果：

A. 第 1 種型式：當行號被執行後會自動列印出來。注意，在行號 30090的迴路FOR：NEXT 可用來控制調整執行速率。上限爲 PDL（0），因此可完全控制執行速率。注意，可利用 paddle 的開關 0 迫使執行暫停。當再開啓開關後便可繼續執行。

B. 第 2 種型式：當程式執行停止後，打入GOTO 32020命令。會顯示第 1 頁的敍述。打入“ C ”可再顯示其它的頁數，打入“ T ”則重新開始測試別的程式，打入“ E ”則回到 BASIC 。注意，縱使您用第 1 種型式追踪程式，您仍可利用第 2 種型式顯示最後 100行的行號。

一範例一

測試程式

```
 0 REM *** REPEATED ***
10 TRACE
30 GOSUB 100 + RND(3) *10
40 FOR I = 1 TO 10: NEXT I
```

```
 50 GOTO 30
100 PRINT "LINE 100":RETURN
110 PRINT "LINE 110":RETURN
120 PRINT "LINE 120":POP
125 NO TRACE:END
 > RUN 30000
 > RUN
```

追踪輸出結果

```
 30 GOSUB 100+RND(3)*10
110 PRINT "LINE 110":RETURN LINE 110
 30 GOSUB 100+RND(3)*10
 40 FOR I=1 TO 10:NEXT I
  0 REM *** REPEATED ***
 50 GOTO 30
 30 GOSUB 100+RND(3)*10
120 PRINT "LINE 120":POP LINE 120
125 NO TRACE:END
 >
```

提示與警告事項

當程式是由按RESET、Control-C、與NOTRACE而結束時，將 TIR 變成非啓動狀態是一個很好的主意。在還沒有執行公用程式的行號 30000 或 32000 之前，不要試着追踪測試程式。

爲了加強即時追踪公用程式的除錯能力，可自由按下按鈕停止執行程式。對行號30090的遲延迴路適當的選擇，可以使您一次追踪一行。當按下按鈕時，打入Control-C 會

使得程式停在30070（STOPPED AT 30070）。此時您便可利用 BASIC 命令列印和改變程式的變數值。打入 CON ，則會繼續執行。

利用公用程式的一些邏輯，您可以做出一些特殊的追踪技巧，例如在偵測到一連串敍述之後便停止執行。假如不想再使用TRACE則可利用CALL 958返回。

追踪程式的執行會使得執行的速度變慢，但是在此時您並不會計較這些。然而，可利用 TRACE/NOTRACE 加快執行速度。同樣的，當碰到遲延迴路如 FOR I = 1 TO 3000 : NEXT I，您可利用Control - C 命令，再打入 I = 2999 與 CON ，便可加快執行速度。

警告＃1：在輸出敍述中， PRINT 敍述不能使用‘＃’字元。因為 TIR 假設‘＃’是追踪順序的開頭。必須將‘＃’移走或是在 PRINT 敍述前後加 NOTRACE/TRACE 一對指述。

警告＃2：在測試程式中的變數名稱不能和第一種型式中所用的變數名稱一樣。在二種型式中，TR 變數名稱一定要是唯一的。

警告＃3：在測試程式中的行號 0 應該是一個 REM 敍述，如前面所述，可避免混淆。當某一行重複執行時會列印出行號 0 。

警告＃4：一旦TRACE開始時，測試程式不能任意的利用CLR或POKE來修改變數指標（$CC，$CD），除非先打入NOTRACE，再重新設立$13、$14；例如100 NOTRACE：CLR：POKE 19，0：POKE 20，0：TRACE即可。

結　　論

設計此程式的動機是爲了增進整數BASIC中TRACE的功能。然而您可以想像在執行每一個敍述後，可以使其它程式取得控制權，而這可能是多元程式規劃（multiprogramming）的核心所在。

```
29970 REM   ************************
29971 REM   *                      *
29972 REM   *  TRACE LIST UTILITY  *
29973 REM   *    BY ALAN G. HILL   *
29974 REM   *                      *
29975 REM   *      TRACE LIST      *
29976 REM   *                      *
29977 REM   *  COPYRIGHT (C) 1981  *
29978 REM   *    MICRO INK, INC.   *
29979 REM   * CHELMSFORD, MA 01824 *
29980 REM   *  ALL RIGHTS RESERVED *
29981 REM   *                      *
29982 REM   ************************
29983 REM
29984 REM
29985 PRINT : PRINT "'RUN 31000' APPEND": PRINT "'RUN 30000' REAL-TIME LIST"
      : PRINT "'RUN 32000' POST-EXEC SETUP"
29986 PRINT "'GOTO 32020' POST-EXEC LIST": VTAB 20: INPUT "'RETURN' WHEN READY
       TO APPEND",A$
29995 GOTO 31000
29998 REM  'RUN 30000' REAL-TIME
30000 NOTRACE : POKE 54,768 MOD 256: POKE 55,768/256: POKE 19,0: POKE 20,
      0: POKE 787,76: POKE 788,211: POKE 789,3: POKE 790,234: CALL -22447

30004 PRINT "ENABLE TRACE IN YOUR PROGRAM": PRINT "AND 'RUN'."
```

```
30005 REM  TRACE VER1.0 11-28-78
30006 REM  TRACE VER1.1 3-6-79
30007 REM  ADD DISK APPEND CAPABILITY
30010 CALL 945: END
30020 EXECLINE=TR(0): IF EXECLINE#0 THEN 30050
30030 IF RRRRR=1 THEN 30070
30040 RRRRR=1: GOTO 30060
30050 RRRRR=0
30060 LIST EXECLINE
30070 IF PEEK (-16287)>127 THEN 30070
30075 IF EXECLINE=0 THEN 30090
30080 FOR JJJJJ=1 TO 150: NEXT JJJJJ
30090 CALL 954: REM   BACK TO TEST PGM
30100 END
31000 DIM A$(30)
31001 VTAB 24
31002 INPUT "APPEND ",A$
31005 IF A$#"" THEN 31030
31010 POKE 0, PEEK (76): POKE 1, PEEK (77): POKE 76, PEEK (202): POKE 77,
       PEEK (203): CALL -3873: POKE 76, PEEK (0): POKE 77, PEEK (1): END
31030 POKE 0, PEEK (76): POKE 1, PEEK (77): POKE 76, PEEK (202): POKE 77,
       PEEK (203): PRINT "LOAD ";A$;",V": POKE 76, PEEK (0): POKE 77, PEEK
      (1)
31031 PRINT "'RUN 30000' REAL-TIME": PRINT "'RUN 32000' POST TIME": END
31999 REM  'RUN 32000' POST-EXEC
32000 POKE 54,768 MOD 256: POKE 55,768/256: POKE 19,0: POKE 20,0: POKE 787
      ,169: POKE 788,127: POKE 789,133: POKE 790,5: CALL -22447
32010 PRINT "TRACE SET UP. ENABLE TRACE IN YOUR PGM": END
32020 NOTRACE : POKE 54,240: POKE 55,253: IF PEEK (20)#0 THEN 32030: PRINT
      "TRACE NOT ON IN YOUR PGM": GOTO 32090
32030 CALL -936: FOR I=100 TO 1 STEP -1: IF TR(I)=-1 THEN 32060
32040 LIST TR(I)
32050 IF PEEK (37)>18 THEN 32090
32060 NEXT I
32070 GOTO 32090
32080 CALL -936: IF I>1 THEN 32060
32090 PRINT : PRINT "C/T/E ?"
32100 KEY= PEEK (-16384): IF KEY<128 THEN 32100: POKE -16368,0: IF KEY=212
       THEN 32000: IF KEY=195 THEN 32080: END
```

註：因爲空間的關係，我們將主要的程式省略，機器語言程式是在磁碟上的 TRACE INTERRUPT 。

4 圖形與遊戲

簡 介

一個多用途的高解析度函數繪圖程式
——— David P. Allen

縮減 APPLE II 高解析度照片所佔的記憶體空間
——— Bob Bishop

一個使 APPLE 的 LIFE 遊戲更有效率的程式
——— Gregory L. Tibbetts

APPLAYER音樂解析器
——— Richard F. Suitor

改進星際戰爭的音響效果
——— William M. Shryock, Jr.

Galacti-Cube
——— Bob Bishop

第四部份簡介

每一本有關APPLE的書都會提到有關機器的娛樂能力。APPLE的二項娛樂能力是繪圖能力與聲音產生能力。這一章介紹一些利用此二種能力的程式，並包括一個太空迷宮遊戲！

Devid Allen的“一個多用途的高解析度函數繪圖程式”，利用高解析度繪圖能力繪出任何使用者定義的函數曲線。Bob Bishop 的“縮減APPLE II 高解析度照片所佔的記憶體空間”，允許使用者利用相同圖形的優點，把在螢幕上影像所佔的記憶體位置縮減。Greg Tibbetts 的“一個使APPLE的LIFE遊戲更有效率的程式”，在“LIFE”模擬時，允許將圖形存入磁碟中。

Dick Suitor 的“Applayer 音樂解析器”，完成一個複雜的音樂產生系統，而沒有用到其它的硬體。William Shryock 的“改進星際戰爭的音響效果”，是另一個聲音產生常規。雖然它的程式比較短，但娛樂效果很高。

最後，Bob Bishop 的“Galacti-Cube ”是一個太空迷宮遊戲，它是由整數BASIC寫的，看你能不能找到出路!

一個多用途的高解析度函數繪圖程式

作者：David P. Allen

APPLE的高解析度功能通常用來繪出各種不同的數學函數。這是一個通用的繪圖程式，可讓使用者很容易的繪出角度從1°到360°的任何函數圖形。程式也有略加修改，可用於Atari。

利用電腦的高解析度繪圖能力來繪出各種不同的數學函數表示式，是一件很有趣的事。在此，我利用APPLE II電腦的高解析度繪圖能力，設計一個能夠繪出各種不同數學函數圖形的程式，願大家共享之。

此程式是在APPLE II 48K RAM與APPLESOFT ROM卡上發展的。整個程式只佔3K又多一點的RAM，主要根據所要繪的函數圖形的複雜度而定。

若沒有APPLESOFT ROM卡，只要將行480的“HGR”改成“HGR2”，即可使用此程式。此情況下，函數公式不會印在螢幕的底端。

此程式時的核心是行號1010，它包含所要繪的函數。在程式中有一個典型函數。當執行時，程式首先定義一些

APPLESOFT BASIC沒有的三角函數與雙曲線函數。接着畫出X軸和Y軸。目前所安排的是將函數從1°畫到360°。可改變行號670和900，便可改變這些限制。此程式會自動取刻度的大小。

當繪圖結束時；程式會印出函數，並等你按Return 鍵。程式接着會印出程式本身所定義的一些函數，它們不包括在APPLESOFT BASIC中。行號1010 會再列印出來，並且游標（ cursor ）將請你再修改函數，以便再繪出其它圖形。

注意事項：不要想要畫出某負值的平方根或log值，否則你會得到一些錯誤消息。但可利用ABS來解決問題。

此程式可能寫得很笨拙。你可在行號1010 前後使用FOR… NEXT 迴路，允許輸入較長的表示式。此程式可能會使你整夜不想睡覺喔！祝愉快！

```
10  REM *************************
12  REM *                       *
14  REM *    FUNCTION PLOTTER   *
16  REM *     DAVID P. ALLEN    *
18  REM *                       *
20  REM *       FNPLOTTER       *
22  REM *                       *
24  REM *  COPYRIGHT (C) 1981   *
26  REM *    MICRO INK, INC.    *
28  REM * CHELMSFORD, MA 01824  *
30  REM *  ALL RIGHTS RESERVED  *
32  REM *                       *
34  REM *************************
140  REM
150  REM
180  REM  THIS PROGRAM PLOTS A
190  REM  CURVE FOR ANY EXPRESSION
200  REM  AS A FUNCTION OF INCREAS-
210  REM  ING ANGLE FROM 1 TO 360
```

```
220  REM  DEGREES.
230  REM  CHANGE LINE 1010 TO A
240  REM  FUNCTION YOU WISH TO
250  REM  PLOT.
260  REM
270  REM
280  REM  *** DEFINE FUNCTIONS ***
290  REM
300  DEF  FN SCH(X) = 2 / ( EXP (X) +  EXP ( - X)): REM  SECH(X)
310  DEF  FN CCH(X) = 2 / ( EXP (X) -  EXP ( - X)): REM  CSCH (X)
320  DEF  FN CTH(X) =  EXP ( - X) / ( EXP (X) -  EXP ( - X)) * 2 + 1:
      REM COTH(X)
330  DEF  FN SEC(X) = 1 /  COS (X): DEF  FN CSC(X) = 1 /  SIN (X): DEF
     FN COT(X) = 1 /  TAN (X)
340  DEF  FN SNH(X) = ( EXP (X) -  EXP ( - X)) / 2: REM  SINH(X)
350  DEF  FN COH(X) = ( EXP (X) +  EXP ( - X)) / 2: REM  COSH(X)
360  DEF  FN TAH(X) =  -  EXP ( - X) / ( EXP (X) +  EXP ( - X)) * 2 + 1:
      REMTANH(X)
370  REM
380  REM
390  REM  ** PLOT GRAPH AXES **
400  REM
410  HOME
420  REM
430  REM  MOVE CURSOR TO BOTTOM
440  REM  LINE.
450  REM
460  VTAB 24
470  REM
480  HGR
490  HCOLOR= 7
500  HPLOT 0,80 TO 279,80
510  HPLOT 0,16 TO 0,143
520  FOR I = 0 TO 279 STEP 70
530  HPLOT I,78 TO I,82: HPLOT 279,78 TO 279,82
540  NEXT I
550  FOR I = 16 TO 144 STEP 16
560  HPLOT 0,I TO 4,I
570  NEXT I
580  REM
590  REM  FLAGS FOR FIRST PLOT
600  REM  AND SCALE.
610  REM
620 F = 0:G = 0
630  REM
640  REM  R1 AND R2 MAY BE SET
650  REM  FOR OTHER LIMITS.
660  REM
670 R1 = 1:R2 = 360
680  REM
690  REM
700  REM  ** BEGIN PLOT **
710  REM
720  REM  CHANGE STEP FOR MORE
730  REM  OR LESS RESOLUTION.
740  REM  IF R1>R2 THEN STEP
750  REM  MUST BE NEGATIVE.
760  REM
770  FOR I = R1 TO R2 STEP .5
780  REM
790  REM  NEXT 3 STEPS ESTABLISH
```

```
800  REM  HORIZONTAL SCALE.
810  REM
820  IF  ABS (R1) >  =  ABS (R2) THEN R =  ABS (R1)
830  IF  ABS (R2) >  =  ABS (R1) THEN R =  ABS (R2)
840  IF G = 0 THEN S = 70 * 4 / R:G = 1
850 X = I:Y = 0
860  REM
870  REM  CONVERTS DEGREES TO
880  REM  RADIANS.
890  REM
900 X = X * 3.14159 / 180
910  REM
920  REM  PREVENTS CRASHING WHEN
930  REM  X=0.
940  REM
950  IF X = 0 THEN X = .00001
960  REM
970  REM
980  REM  NEXT LINE DESCRIBES
990  REM  FUNCTION TO BE PLOTTED
1000  REM
1010 Y1 =  SIN (X) +  COS (2 * X)
1020 Y = Y + Y1
1030 Y = Y * 20
1040  REM
1050  REM  SCALES X
1060  REM
1070 X = I * S
1080  REM
1090  REM  RELATES PLOT TO X AXIS
1100  REM
1110 Y =  - Y + 80
1120  REM
1130  REM  SUBROUTINE PREVENTS
1140  REM  OFF-SCALE CRASHING.
1150  REM
1160  GOSUB 1830
1170  REM
1180  REM  PLOTS FIRST POINT.
1190  REM
1200  IF F = 0 THEN  HPLOT X,Y:F = 1
1210  HPLOT  TO X,Y
1220  NEXT I
1230  PRINT : LIST 1010
1240  REM
1250  REM  BLANKS OUT LINE #
1260  REM  AFTER LISTING
1270  REM  LINE 1010.
1280  REM
1290  POKE 1616,160: POKE 1617,160: POKE 1618,160: POKE 1619,160
1300  REM
1310  REM  WAITING FOR YOUR PLEASURE!
1320  REM  PUNCH 'RETURN'
1330  REM  TO CONTINUE!
1340  REM
1350  POKE  - 16368,0: WAIT  - 16384,128
1360  REM
1370  REM
```

```
1380  REM  THROWS PREVIOUS KEYSTROKE
1390  REM  AWAY WITH
1400  REM  'GET Z$'!
1410  REM
1420  GET Z$
1430  REM
1440  REM  CLEAR SCREEN AND
1450  REM  PRINT FUNCTIONS FOR
1460  REM  REMINDER.
1470  REM
1480  TEXT : HOME
1490  PRINT  TAB( 9);"SECANT = FN SEC(X)"
1500  PRINT  TAB( 9);"COSEC = FN CSC(X)"
1510  PRINT  TAB( 9);"COTAN = FN COTAN(X)"
1520  PRINT  TAB( 9);"SINH = FN SNH(X)"
1530  PRINT  TAB( 9);"COSH = FN COH(X)"
1540  PRINT  TAB( 9);"TANH = FN TAH(X)"
1550  PRINT  TAB( 9);"SECH = FN SCH(X)"
1560  PRINT  TAB( 9);"CSCH = FN CCH(X)"
1570  PRINT  TAB( 9);"COTH = FN CTH(X)"
1580  REM
1590  REM  NOW WE SET UP LINE
1600  REM  1010 FOR EDITING.
1610  REM  'POKE 32, 2' MOVES
1620  REM  MARGIN SO CURSOR CAN
1630  REM  FIT IN FRONT.
1640  REM
1650  VTAB (12)
1660  PRINT " CHANGE LINE 1010 AS DESIRED AND"
1670  PRINT "RUN AGAIN!"
1680  POKE 32,2
1690  LIST 1010
1700  REM
1710  REM  NOW WE RESTORE MARGIN
1720  REM  AND MOVE CURSOR IN
1730  REM  FRONT OF LINE #.
1740  REM
1750  POKE 32,0
1760  POKE 37,13: POKE 36,0
1770  REM
1780  END
1790  REM
1800  REM  SCALE ANTI-CRASHING
1810  REM  SUBROUTINE.
1820  REM
1830  IF X < 0 THEN X = 0
1840  IF X > 279 THEN X = 279
1850  IF Y < 0 THEN Y = 0
1860  IF Y > 159 THEN Y = 159
1870  RETURN
```

縮減APPLE Ⅱ 高解析度照片所佔的記憶體空間

作者：Bob Bishop

每一個APPLE所有者都知道APPLE能利用高解析度圖形畫出一些漂亮的照片。本程式將提供一些技巧，使得處理照片能產生一些特殊的效果，並減少儲存空間。

幾乎大部份的APPLE所有者都看過APPLE II 能利用高解析度圖形繪出一些數位照片。這些照片是由192×280個相同明暗度的點陣列所組成的。集合這些點成一整體，便可形成有明暗度的照片。我和Bill Atkinson 共同製作一些這種照片，可由APPLE SOFTWARE BANK取得。通常在一片標準13- 磁扇（ sector ）的磁碟上可存11 張照片。

在利用高解析度顯示照片之前，必須先將高解析度照片存入二個高解析度顯示區其中的一個。第一區，$2000－$3FFF，稱做主顯示緩衝區；第二區，$4000－$5FFF，稱做次顯示緩衝區。明顯的，每個顯示區各佔 8K位元組的長度。因此，高解析度圖形通常儲存爲 8K大小的資料，

剛好存在顯示緩衝區。但是我們一定要那樣存嗎？

假如你對一個高解析度照片仔細近看，你會發現到，在照片中有些小地區根本是一樣的。例如，高解析度的顯示通常是含有純白或純黑的地區。在一張顯示的照片中，某些灰色的圖形也可能和其它灰色圖形類似。所以明顯地，高解析度照片含有一些多餘相同（ redundancy ）的圖形。假如有辦法將這些多餘相同的圖形去掉，則一張高解析度照片的儲存空間便可少於 8K 位元組。

假設我們將顯示圖形分割成許多小的矩形方塊，每個方塊有 7 個位元寬， 8 個位元高。因此一張照片含有 24 行照片元素（像點（ pixels ）），每行有 40 個像點。（這好像 APPLE II TEXT 型式中的 24 列，而每列有 40 行！）因此，定義一張高解析度照片所需要的總像點數爲 40 × 24 = 960。然而，如果有多餘相同的圖形存在，則就不需要 960 個像點。

爲了說明這些概念，我採用 Atkinson 的 LADY BE GOOD照片，如圖 1 所示，並且寫一個程式將所有不同的像點取出。我發現在 960 個像點中只有 662 個像點是唯一的。這表示幾乎有⅓張照片是多餘的！

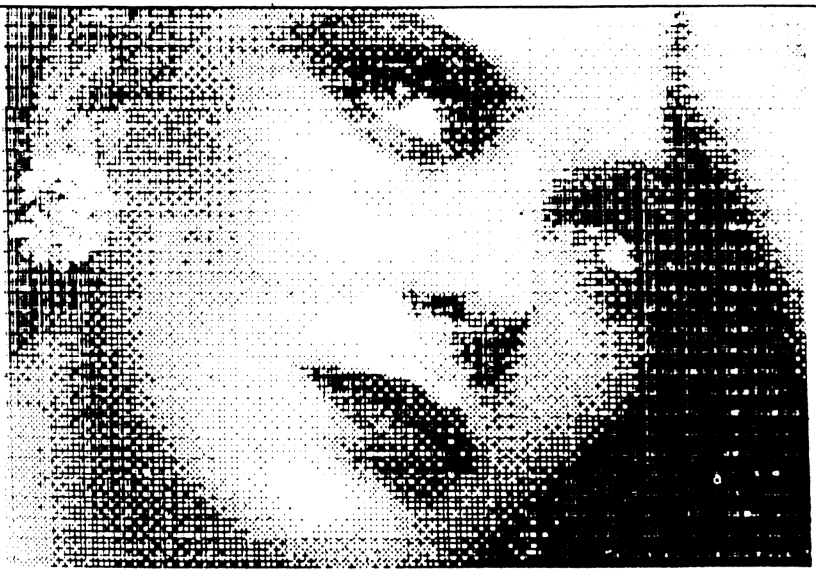


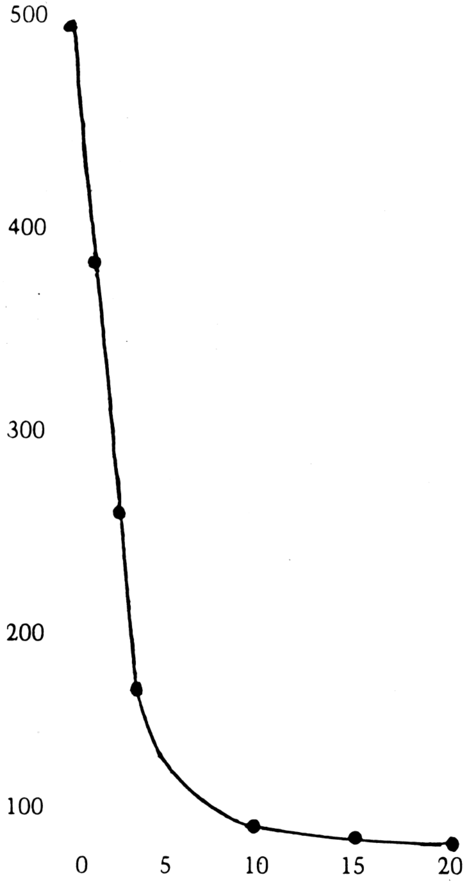
圖1：（最大誤差／像點＝0）

下一個想到的問題是：這662個唯一的像點，它們是如何的“唯一”？有沒有可能二個或很多像點幾乎相同，除了一個或二個點（dots）不同？如果是這樣，則可以將它們看成是相同，因爲得出的照片結果很難看出差異。

爲了測試此可行性，我修改程式使得取出的像點，其所差的點數大於一固定值（最大誤差／像點）。表1就是執行的結果。假如我們在任何一個像點中，最多允許1點的誤差，則我們只需要用492個像點便可定義此照片，這僅僅是原來960個像點的一半。當每個像點的誤差愈大時，所需要的

像點就愈少，直到誤差點數為 28 點。此情況下，我們允許一半的點數誤差。所以若是最大誤差點數為 28 ，則照片可由二種像點組成，即黑像點和白像點。

表 1：最大誤差 / 像點



max errors per pixel	number of pixels
0	662
1	492
2	334
3	245
4	178
5	135
6	100
7	75
8	53
9	44
10	34
11	29
12	24
13	18
14	15
15	11
16	10
17	7
18	7
19	6
20	5
22	4
24	4
26	4
28	2

假設我們現在要從取出的像點重新組合成原來的照片。明顯的，像點愈少，則所組合的照片愈差。圖 2 到圖 5 分別顯示最大誤差/像點為 3，7，14 與 28 的照片。它們所用的像點數分別為 245，75，15，與 2。注意，圖 1 和圖 2 看起來並沒有什麼差異。主要的好處是圖 2 照片所佔的記憶體空間小於 3K 位元組！（245 個像點，每個像點佔 8 個位元組，再加上 960 個位元組定義那個像點要放在那裏）。

因此很明顯的，假如你願意影像品質有點失真，你可以將 8K 的高解析度照片儲存在小於 8K 位元組的記憶體中。利用此原則，我在一片磁碟上可儲存 33 張照片影像。

縮減儲存空間的程式

後面所附的程式與一些相關的資料表格需要在最少 32K 位元組的 APPLE II 上執行。此常規包含二個部份——"分析"部份，與"綜合"部份。

分析常規（$0B00）將搜尋高解析度的主顯示緩衝區（$2000－$3FFF），並且對每個像點與現時的像點表（從 $0600 開始）比較。假如在指定的最大誤差（位址 $10）內，某個像點和像點表中的某像點相配合，則再取下一個像點比較。假如表中沒有可配合的，則將像點加入像點表中，再繼續比較下一像點。

綜合常規（$0B80）剛好和分析常規相反。它首先把

主緩衝區的每個像點和像點表中的每個像點比較，找到最好的配合。然後將此像點放入次緩衝區相對的位置，利用像點表中的像點找出最接近原圖的像點。

此常規非常容易使用。將照片載入 $2000 － $3FFF 中，設立像點的最大誤差的 $10 ，再召用 $0B00 的常規。當執行完時，位址 $07 與 $08 存着取出的像點數，總數＝1＋（ $07 的值）＋ 40 ＊（ $08 的值）。



圖 2：（最大誤差／像點＝ 3 ）

要由取出的像點組合成照片，只要叫用 $0B80 的常規。當常規執行完時，重新組合的照片會存在高解析度的次緩衝區（ $4000 － $5FFF ）。

假如你有一個48K的APPLE與磁碟，你可以使用後面所附的BASIC程式。此程式將很容易的使用。首先程式會列出一些指令讓你選擇。

L——將照片由磁碟載入到高解析度的主緩衝區。

1——將在高解析度主緩衝區內的照片顯示出來。

2——將在高解析度次緩衝區內的照片顯示出來。

A——分析原照片（產生像點表）。

S——利用現時的像點表組合原照片。

D——發出磁碟命令。

X——將縮減的照片傳到2號磁碟機。

每一個選擇都不必按Return；只要按相對的字元即可。當按下“L”，程式會問你所要載入的檔案名稱。當按下“A”，程式會問你像點的最大誤差（在此就需要按Return）。按下“D”，程式會打出“：”字元，允許你發出磁碟命令。若想回到BASIC程式，就打空命令(即只按Return)。打入“X”程式會把縮減的照片（960個位元組）與它相對的像點表（最多到2K位元組）存入磁碟檔中。

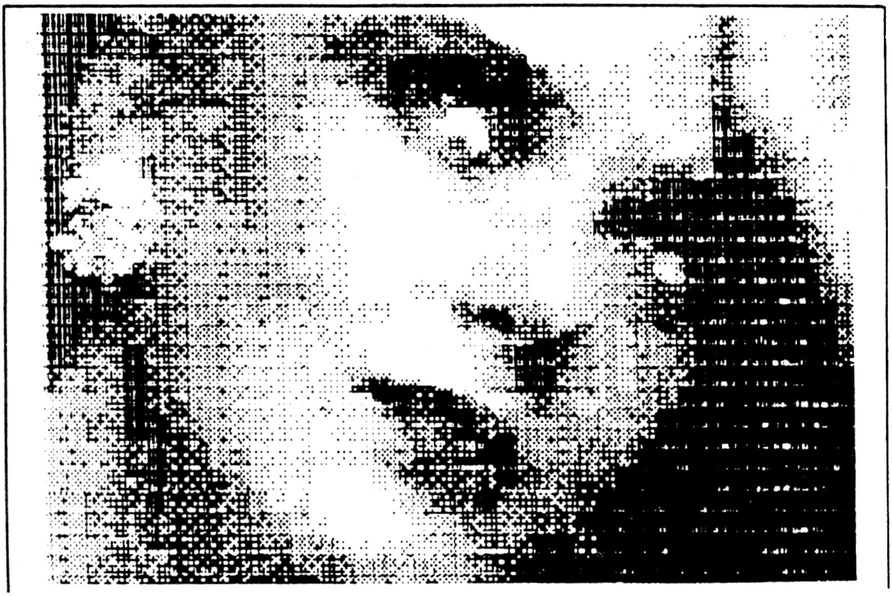


圖 3：（最大誤差/像點= 7）

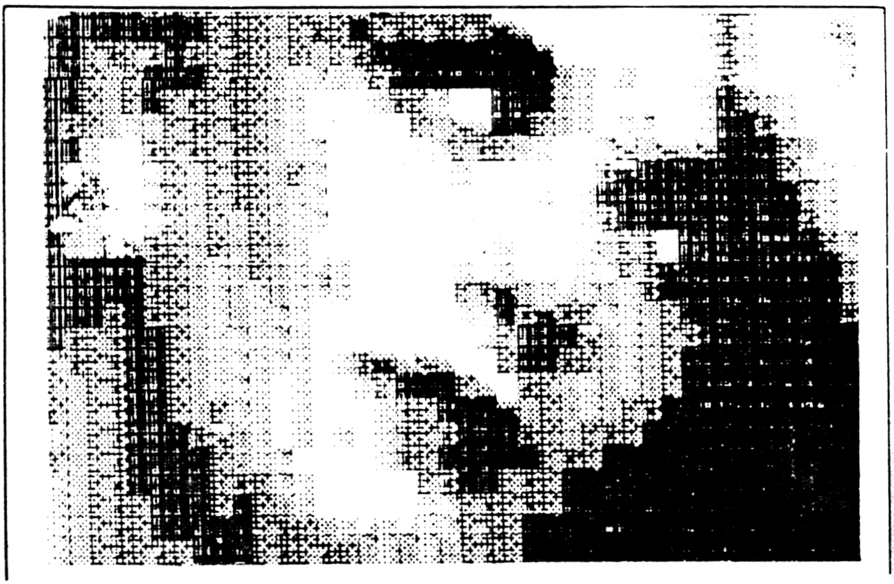


圖 4：（最大誤差/像點= 14）

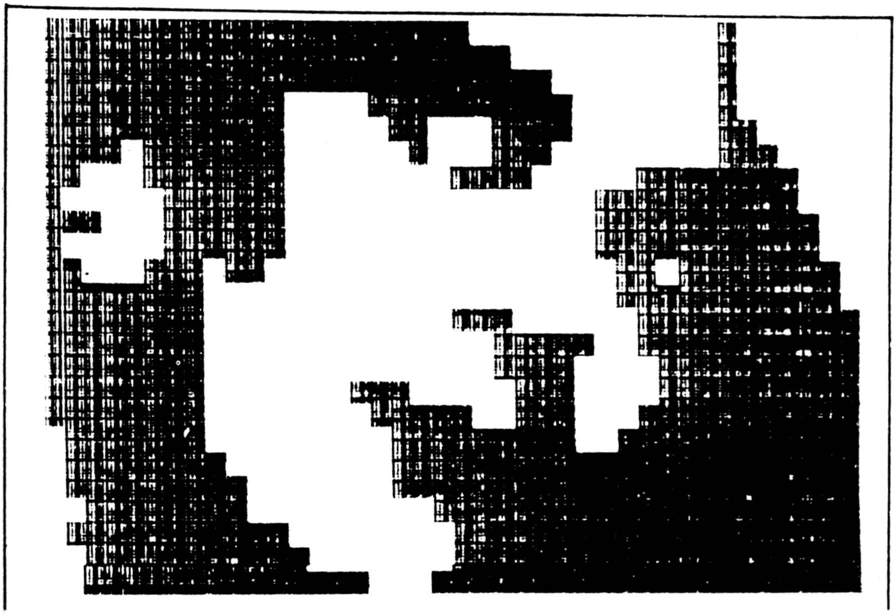


圖 5：（最大誤差/像點= 28 ）

雖然此方法在此做起來效果還不錯，但並不一定是最好的。例如，我選擇一個像點為 7 × 8 個點也是隨意的。若選擇大一點（或小一點）有沒有可能好一點？或者，從某 n 個像點組成的照片，是不是可找出另一組 n 個像點，再組成較好的照片？

```
0C00- 00 00 00 00 00 00 00 00
0C08- 80 80 80 80 80 80 80 80
0C10- 00 00 00 00 00 00 00 00
0C18- 80 80 80 80 80 80 80 80
0C20- 00 00 00 00 00 00 00 00
0C28- 80 80 80 80 80 80 80 80
0C30- 00 00 00 00 00 00 00 00
0C38- 80 80 80 80 80 80 80 80
0C40- 28 28 28 28 28 28 28 28
0C48- A8 A8 A8 A8 A8 A8 A8 A8
0C50- 28 28 28 28 28 28 28 28
0C58- A8 A8 A8 A8 A8 A8 A8 A8
0C60- 28 28 28 28 28 28 28 28
0C68- A8 A8 A8 A8 A8 A8 A8 A8
0C70- 28 28 28 28 28 28 28 28
0C78- A8 A8 A8 A8 A8 A8 A8 A8
0C80- 50 50 50 50 50 50 50 50
0C88- D0 D0 D0 D0 D0 D0 D0 D0
0C90- 50 50 50 50 50 50 50 50
0C98- D0 D0 D0 D0 D0 D0 D0 D0
0CA0- 50 50 50 50 50 50 50 50
0CA8- D0 D0 D0 D0 D0 D0 D0 D0
0CB0- 50 50 50 50 50 50 50 50
0CB8- D0 D0 D0 D0 D0 D0 D0 D0
```

```
0D00- 20 24 28 2C 30 34 38 3C
0D08- 20 24 28 2C 30 34 38 3C
0D10- 21 25 29 2D 31 35 39 3D
0D18- 21 25 29 2D 31 35 39 3D
0D20- 22 26 2A 2E 32 36 3A 3E
0D28- 22 26 2A 2E 32 36 3A 3E
0D30- 23 27 2B 2F 33 37 3B 3F
0D38- 23 27 2B 2F 33 37 3B 3F
0D40- 20 24 28 2C 30 34 38 3C
0D48- 20 24 28 2C 30 34 38 3C
0D50- 21 25 29 2D 31 35 39 3D
0D58- 21 25 29 2D 31 35 39 3D
0D60- 22 26 2A 2E 32 36 3A 3E
0D68- 22 26 2A 2E 32 36 3A 3E
0D70- 23 27 2B 2F 33 37 3B 3F
0D78- 23 27 2B 2F 33 37 3B 3F
0D80- 20 24 28 2C 30 34 38 3C
0D88- 20 24 28 2C 30 34 38 3C
0D90- 21 25 29 2D 31 35 39 3D
0D98- 21 25 29 2D 31 35 39 3D
0DA0- 22 26 2A 2E 32 36 3A 3E
0DA8- 22 26 2A 2E 32 36 3A 3E
0DB0- 23 27 2B 2F 33 37 3B 3F
0DB8- 23 27 2B 2F 33 37 3B 3F
```

```
1000- 00 01 01 02 01 02 02 03
1008- 01 02 02 03 02 03 03 04
1010- 01 02 02 03 02 03 03 04
1018- 02 03 03 04 03 04 04 05
1020- 01 02 02 03 02 03 03 04
1028- 02 03 03 04 03 04 04 05
1030- 02 03 03 04 03 04 04 05
1038- 03 04 04 05 04 05 05 06
1040- 01 02 02 03 02 03 03 04
1048- 02 03 03 04 03 04 04 05
1050- 02 03 03 04 03 04 04 05
1058- 03 04 04 05 04 05 05 06
1060- 02 03 03 04 03 04 04 05
1068- 03 04 04 05 04 05 05 06
1070- 03 04 04 05 04 05 05 06
1078- 04 05 05 06 05 06 06 07
```

```
 1 REM ***********************
 2 REM *                     *
 3 REM * PICTURE COMPRESSION *
 4 REM *  BY ROBERT BISHCP   *
 5 REM *                     *
 6 REM *      COMPRESS       *
 7 REM *                     *
 8 REM * COPYRIGHT (C) 1981  *
 9 REM *   MICRO INK, INC.   *
10 REM * CHELMSFORD, MA 01824 *
11 REM * ALL RIGHTS RESERVED *
12 REM *                     *
13 REM ***********************
14 REM
```

```
  15 DIM A$(40)
  20 ANAL=11*256:SYN=ANAL+128:PRESS=4096+2*256+8*16
  30 FLAG=0:XFLAG=0
  50 PRINT "BLOAD PIXEL STUFF"
 100 CALL -936: POKE -16300,0: POKE -16303,0
 110 TAB 17: PRINT "M E N U"
 120 TAB 17: PRINT "-------": PRINT
 130 PRINT : PRINT "   L - LOAD PICTURE FROM DISK"
 140 PRINT : PRINT "   A - ANALYZE PICTURE INTO PIXELS"
 150 PRINT : PRINT "   S - SYNTHESIZE PICTURE FROM PIXELS"
 160 PRINT : PRINT "   1 - DISPLAY ORIGINAL PICTURE"
 170 PRINT : PRINT "   2 - DISPLAY SYNTHESIZED PICTURE"
 180 PRINT : PRINT "   D - ISSUE DISK COMMANDS"
 190 PRINT : PRINT "   X - SAVE COMPRESSED PICTURE TO DISK"
 195 VTAB 20: PRINT "SELECTION: "
 200 REM READ KEYBOARD
 210 CHAR= PEEK (-16384)
 220 IF CHAR<128 THEN 210
 230 POKE -16384+16,0
 300 ID=0
 310 IF CHAR= ASC("L") THEN ID=1
 320 IF CHAR= ASC("A") THEN ID=2
 330 IF CHAR= ASC("S") THEN ID=3
 340 IF CHAR= ASC("1") THEN ID=4
 350 IF CHAR= ASC("2") THEN ID=5
 360 IF CHAR= ASC("D") THEN ID=6
 370 IF CHAR= ASC("X") THEN ID=7
 400 IF ID=0 THEN 100
 500 GOTO 1000*ID
1000 VTAB 20: TAB 12: CALL -958: PRINT "LOAD PICTURE"
1005 POKE -16300,0: POKE -16303,0
1010 VTAB 22: INPUT "FILE NAME: ",A$
1015 IF A$="" THEN 100
1020 VTAB 22: PRINT "BLOAD ";A$;",A$2000,D1"
1050 GOTO 100
2000 VTAB 20: TAB 12: CALL -958: PRINT "ANALYZE PICTURE"
2005 POKE -16300,0: POKE -16303,0
2010 VTAB 22: INPUT "MAX ERRORS/PIXEL:",MAXERR
2020 POKE 16,MAXERR: CALL ANAL
2025 FLAG=1:XFLAG=0:NUMBER=40* PEEK (8)+ PEEK (7)+1
2030 VTAB 22: PRINT "THERE ARE ";NUMBER;" PIXELS WITH MAX ERROR = ";MAXERR
2035 POKE -16384+16,0
2040 IF PEEK (-16384)<128 THEN 2040
2050 GOTO 100
3000 VTAB 20: TAB 12: PRINT "SYNTHESIZE PICTURE"
3005 POKE -16300,0: POKE -16303,0: VTAB 22: CALL -958
3010 FOR K=1 TO 500: NEXT K
3020 IF FLAG THEN 3050
3030 VTAB 22: PRINT "THERE ARE NO PIXELS DEFINED YET!"
3040 GOTO 3060
3050 CALL SYN
3055 XFLAG=1
3060 POKE -16384+16,0
3070 IF PEEK (-16384)<128 THEN 3070
3080 IF PEEK (-16384)= ASC("1") THEN 210
3085 IF PEEK (-16384)= ASC("2") THEN 210
3090 GOTO 100
4000 POKE -16304,0: POKE -16302,0: POKE -16300,0: POKE -16297,0
4050 GOTO 200
5000 POKE -16304,0: POKE -16302,0: POKE -16299,0: POKE -16297,0
5050 GOTO 200
6000 VTAB 20: TAB 12: CALL -958: PRINT "DISK COMMAND"
6005 POKE -16300,0: POKE -16303,0
```

```
6010 VTAB 22: INPUT ":",A$
6015 IF A$="" THEN 100
6020 VTAB 22: TAB 2: PRINT "";A$
6030 PRINT : PRINT : PRINT
6040 GOTO 6010
7000 VTAB 20: TAB 12: CALL -958: PRINT "SAVE COMPRESSED PICTURE"
7005 POKE -16300,0: POKE -16303,0
7010 IF XFLAG THEN 7025
7015 VTAB 22: PRINT "NO PICTURE HAS BEEN SYNTHESIZED YET!"
7020 GOTO 7040
7025 IF NUMBER<=256 THEN 7060
7030 VTAB 22: PRINT "THERE ARE TOO MANY (";NUMBER;") PIXELS"
7040 POKE -16384+16,0
7045 IF PEEK (-16384)<128 THEN 7045
7050 GOTO 100
7060 VTAB 22: INPUT "FILE NAME: ",A$
7065 IF A$="" THEN 100
7070 CALL PRESS
7080 VTAB 22: PRINT "BSAVE ";A$;",A$8000,L";960+2+8*NUMBER;",D2"
7090 GOTO 100
```

```
0800    ;*************************
0800    ;*                       *
0800    ;*   PICTURE COMPRESSION *
0800    ;*      ROBERT BISHOP    *
0800    ;*                       *
0800    ;*        PICT COMP      *
0800    ;*                       *
0800    ;*   COPYRIGHT (C) 1981  *
0800    ;*     MICRO INK, INC.   *
0800    ;* CHELMSFORD, MA 01824  *
0800    ;*  ALL RIGHTS RESERVED  *
0800    ;*                       *
0800    ;*************************
0800    ;
0800    ;
0800    ;
0800    XAT     EPZ $0000
0800    YAT     EPZ $0001
0800    ZAT     EPZ $0002
0800    XTO     EPZ $0003
0800    YTO     EPZ $0004
0800    ZTO     EPZ $0005
0800    SCOR    EPZ $0006
0800    XMAX    EPZ $0007
0800    YMAX    EPZ $0008
0800    XTMP    EPZ $0009
0800    YTMP    EPZ $000A
0800    BEST    EPZ $000B
0800    AT      EPZ $000C
0800    TO      EPZ $000E
0800    ERR     EPZ $0010
0800    XIN     EPZ $0011
0800    YIN     EPZ $0012
0800    PROD    EPZ $0013
0800    HGRL    EQU $0C00
0800    HGRH    EQU $0D00
0800    BITS    EQU $1000
0800    BELL    EQU $FF3A
0800    ;
0800    ;
0800            ORG $800
```

```
0B00                        OBJ $800
0B00                ;
0B00 209311         BILD    JSR INIT
0B03 A900                   LDA #$00
0B05 8500                   STA XAT
0B07 8501                   STA YAT
0B09 A901                   LDA #$01
0B0B 8502                   STA ZAT
0B0D A903                   LDA #$03
0B0F 8505                   STA ZTO
0B11 A900           BLUP    LDA #$00
0B13 8503                   STA XTO
0B15 8504                   STA YTO
0B17 202311         LUPE    JSR COMP
0B1A A510                   LDA ERR
0B1C C506                   CMP SCOR
0B1E B01F                   BCS GOOD
0B20 A503                   LDA XTO
0B22 C507                   CMP XMAX
0B24 D006                   BNE NEXT
0B26 A504                   LDA YTO
0B28 C508                   CMP YMAX
0B2A F005                   BEQ OVER
0B2C 20F10B         NEXT    JSR NUTO
0B2F D0E6                   BNE LUPE
0B31 20F10B         OVER    JSR NUTO
0B34 200011                 JSR MOVE
0B37 A503                   LDA XTO
0B39 8507                   STA XMAX
0B3B A504                   LDA YTO
0B3D 8508                   STA YMAX
0B3F E600           GOOD    INC XAT
0B41 A500                   LDA XAT
0B43 C928                   CMP #$28
0B45 D0CA                   BNE BLUP
0B47 A900                   LDA #$00
0B49 8500                   STA XAT
0B4B E601                   INC YAT
0B4D A501                   LDA YAT
0B4F C918                   CMP #$18
0B51 D0BE                   BNE BLUP
0B53 4C3AFF                 JMP BELL
0B56                ;
0B56                ; RECONSTRUCTION
0B56                ;
0B80                        ORG $B80
0B80                        OBJ $880
0B80                ;
0B80 A900           RCON    LDA #$00
0B82 8D50C0                 STA $C050
0B85 8D52C0                 STA $C052
0B88 8D55C0                 STA $C055
0B8B 8D57C0                 STA $C057
0B8E 8503                   STA XTO
0B90 8504                   STA YTO
0B92 A903                   LDA #$03
0B94 8502                   STA ZAT
0B96 A9FF           RLUP    LDA #$FF
0B98 850B                   STA BEST
0B9A A900                   LDA #$00
0B9C 8500                   STA XAT
0B9E 8501                   STA YAT
```

```
0BA0 A901                 LDA #$01
0BA2 8505                 STA ZTC
0BA4 202311       LOOP    JSR COMP
0BA7 A506                 LDA SCCR
0BA9 C50B                 CMP BEST
0BAB B00A                 BCS CONT
0BAD 850B                 STA BEST
0BAF A500                 LDA XAT
0BB1 8509                 STA XTMP
0BB3 A501                 LDA YAT
0BB5 850A                 STA YTMP
0BB7 A500         CONT    LDA XAT
0BB9 C507                 CMP XMAX
0BBB D006                 BNE INC
0BBD A501                 LDA YAT
0BBF C508                 CMP YMAX
0BC1 F010                 BEQ SEND
0BC3 E600         INC     INC XAT
0BC5 A500                 LDA XAT
0BC7 C928                 CMP #$28
0BC9 D0D9                 BNE LOOP
0BCB A900                 LDA #$00
0BCD 8500                 STA XAT
0BCF E601                 INC YAT
0BD1 D0D1                 BNE LOOP
0BD3 A509         SEND    LDA XTMP
0BD5 8500                 STA XAT
0BD7 A50A                 LDA YTMP
0BD9 8501                 STA YAT
0BDB A902                 LDA #$02
0BDD 8505                 STA ZTC
0BDF 200012               JSR STOR
0BE2 200011               JSR MOVE
0BE5 20F10B               JSR NUTO
0BE8 A504                 LDA YTC
0BEA C918                 CMP #$18
0BEC D0A8                 BNE RLUP
0BEE 4C3AFF               JMP BELL
0BF1 E603         NUTO    INC XTO
0BF3 A503                 LDA XTC
0BF5 C928                 CMP #$28
0BF7 D006                 BNE RET
0BF9 A900                 LDA #$00
0BFB 8503                 STA XTC
0BFD E604                 INC YTC
0BFF 60           RET     RTS
0C00                      ;
0C00                      ; MOVE A PIXEL FROM XAT,YAT,ZAT
0C00                      ; TO XTO,YTO,ZTO....
0C00                      ;
1100                              ORG $1100
1100                              OBJ $E00
1100                      ;
1100 8A           MOVE    TXA
1101 48                   PHA
1102 98                   TYA
1103 48                   PHA
1104 205411               JSR PREP
1107 A400         MLUP    LDY XAT
1109 B100                 LDA (AT),Y
110B A403                 LDY XTC
```

```
110D 910E             STA (TO),Y
110F A50D             LDA AT+1
1111 6904             ADC #$04
1113 850D             STA AT+1
1115 A50F             LDA TO+1
1117 6904             ADC #$04
1119 850F             STA TO+1
111B CA               DEX
111C D0E9             BNE MLUP
111E 68               PLA
111F A8               TAY
1120 68               PLA
1121 AA               TAX
1122 60               RTS
1123            ;
1123            ; COMPARE PIXEL AT XAT,YAT,ZAT
1123            ; TO XTO,YTO,ZTO
1123            ;
1123 8A         COMP  TXA
1124 48               PHA
1125 98               TYA
1126 48               PHA
1127 205411           JSR PREP
112A A900             LDA #$00
112C 8506             STA SCOR
112E A400       CLUP  LDY XAT
1130 B10C             LDA (AT),Y
1132 A403             LDY XTO
1134 510E             EOR (TO),Y
1136 297F             AND #$7F
1138 A8               TAY
1139 B90010           LDA BITS,Y
113C 6506             ADC SCOR
113E 8506             STA SCOR
1140 A50D             LDA AT+1
1142 6904             ADC #$04
1144 850D             STA AT+1
1146 A50F             LDA TO+1
1148 6904             ADC #$04
114A 850F             STA TO+1
114C CA               DEX
114D D0DF             BNE CLUP
114F 68               PLA
1150 A8               TAY
1151 68               PLA
1152 AA               TAX
1153 60               RTS
1154            ;
1154 A502       PREP  LDA ZAT
1156 6A               ROR
1157 6A               ROR
1158 6A               ROR
1159 6A               ROR
115A 2960             AND #$60
115C 850D             STA AT+1
115E A505             LDA ZTO
1160 6A               ROR
1161 6A               ROR
1162 6A               ROR
1163 6A               ROR
1164 2960             AND #$60
```

```
1166 850C          STA  TO+1
1168 A501          LDA  YAT
116A 0A            ASL
116B 0A            ASL
116C 0A            ASL
116D AA            TAX
116E BD000C        LDA  HGRL,X
1171 850C          STA  AT
1173 BD000D        LDA  HGRH,X
1176 291F          AND  #$1F
1178 650D          ADC  AT+1
117A 850D          STA  AT+1
117C A504          LDA  YTO
117E 0A            ASL
117F 0A            ASL
1180 0A            ASL
1181 AA            TAX
1182 BD000C        LDA  HGRL,X
1185 850E          STA  TO
1187 BD000D        LDA  HGRH,X
118A 291F          AND  #$1F
118C 650F          ADC  TO+1
118E 850F          STA  TO+1
1190 A208          LDX  #$08
1192 60            RTS
1193               ;
1193 20C00C   INIT JSR  $0CC0
1196 A97F          LDA  #$7F
1198 8D0160        STA  $6001
119B 8D0164        STA  $6401
119E 8D0168        STA  $6801
11A1 8D016C        STA  $6C01
11A4 8D0170        STA  $7001
11A7 8D0174        STA  $7401
11AA 8D0178        STA  $7801
11AD 8D017C        STA  $7C01
11B0 A900          LDA  #$C0
11B2 8508          STA  YMAX
11B4 A901          LDA  #$01
11B6 8507          STA  XMAX
11B8 60            RTS
11B9
1200               ORG  $1200
1200 98       STCR TYA
1201 48            PHA
1202 A503          LDA  XTO
1204 8511          STA  XIN
1206 A504          LDA  YTO
1208 8512          STA  YIN
120A 202C12        JSR  X40
120D A513          LDA  PROD
120F 850E          STA  TO
1211 18            CLC
1212 A514          LDA  PROD+1
1214 6980          ADC  #$80
1216 850F          STA  TO+1
1218 A500          LDA  XAT
121A 8511          STA  XIN
121C A501          LDA  YAT
121E 8512          STA  YIN
1220 202C12        JSR  X40
1223 A513          LDA  PROD
```

```
1225 A0C0              LDY #$00
1227 910E              STA (TO),Y
1229 68                PLA
122A A8                TAY
122B 60                RTS
122C A512        X40   LDA YIN
122E 8513              STA PROD
1230 A900              LDA #$C0
1232 8514              STA PROD+1
1234 0613              ASL PROD
1236 2614              ROL PROD+1
1238 0613              ASL PROD
123A 2614              ROL PROD+1
123C 0613              ASL PROD
123E 2614              ROL PROD+1
1240 A513              LDA PROD
1242 0613              ASL PROD
1244 2614              ROL PROD+1
1246 0613              ASL PROD
1248 2614              ROL PROD+1
124A 6513              ADC PROD
124C 8513              STA PROD
124E A514              LDA PROD+1
1250 6900              ADC #$00
1252 8514              STA PROD+1
1254 A513              LDA PROD
1256 6511              ADC XIN
1258 8513              STA PROD
125A A514              LDA PROD+1
125C 6900              ADC #$C0
125E 8514              STA PROD+1
1260 60                RTS
                       END

***** END OF ASSEMBLY


          *************************
          *                       *
          * SYMBOL TABLE -- V 1.5 *
          *                       *
          *************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

XAT    000C  YAT    0001  ZAT    0002  XTO    0003  YTO    0004  ZTO    0005
SCOR   0006  XMAX   0007  YMAX   0008  XTMP   0009  YTMP   000A  BEST   000B
AT     000C  TO     000E  ERR    0010  XIN    0011  YIN    0012  PROD   0013


** ABSOLUTE VARABLES/LABELS

HGRL   0C00  HGRH   0D00  BITS   1000  BELL   FF3A  BILD   0B00  BLUP   0B11
LUPE   0B17  NEXT   0B2C  OVER   0B31  GOOD   0B3F  RCON   0B80  RLUP   0B96
LOOP   0BA4  CONT   0BB7  INC    0BC3  SEND   0BD3  NUTO   0BF1  RET    0BFF
MOVE   1100  MLUP   1107  COMP   1123  CLUP   112E  PREP   1154  INIT   1193
STOR   1200  X40    122C

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0172
```

一個使APPLE的LIFE遊戲更有效率的程式

作者：Gregory L. Tibbetts

利用此程式可以使得LIFE遊戲更簡單，此程式將提供圖形的平移、旋轉、與反轉。

John Conway的LIFE 遊戲是最大的一個電腦模擬(simulation)程式。因爲我已經有6個型式的程式，並且玩過很多次，在執行時都大不相同。但每一個都有一個嚴重的缺點，就是每一次要執行時，都要做一些單調的工作把2組到200組的座標打進去。我曾經看過許多用來取圖形座標的程式——非常複雜。然而由於LIFE的喜愛者有時會在相同的位置輸入相同方位的圖形二次，且他們也喜歡合併圖形以達到交談(interactive)的效果，所以這些程式都不適用。有一個系統想利用紙帶存120個以上的圖形，不同方位的圖形最多重複有8次，並且都有記號和目錄。

由於我自己也懶惰(最後，我買一部電腦把我自己的工作存起來)，我決定我需要一些簡單的程式，將圖形命名並存入磁碟中，並且允許我呼叫它們出來並顯示在螢幕的任何位置，任何方位，以及合併其它圖形。我的目的，也就是本

文的目的，就是使LIFE能更容易使用。

我寫的常規是建立在Dick Suitor寫的“LIFE for Your Apple”，它是一個很好的機器語言／整數BASIC語言的混合程式。它出現在MICRO雜誌的Volume 1。它可能是最好的一個，它能夠變化生長的速度，在一選定的地區內設定隨機的活細胞，並且使用彩色的對比顯示細胞的發展情形。

我的第一個工作是找一個方法儲存與取出圖形。很明顯的解法就是將X，Y座標值存在一個順序檔案中。使得圖形能夠重新定位，使得儲存的座標和螢幕座標沒有關係。我選用的方法是，在輸入圖形之前任意選一個圖形的中心點。所以當每組座標打入時，將打入的X，Y值減掉中心點的X，Y值。結果就是一組修正過的X，Y值，正的或負的是和中心點的位置有關，並且完全和它們現時的螢幕位置無關。當要重新定圖形的位置時，只要把圖形叫出來並改變中心點的位置即可。

這種方法和APPLE的螢幕座標配合使用時，會產生一些不規則變化，這種變化很重要，下面再談。在正常的座標系統下，X增加時會向右移，Y增加時會往上移。在APPLE II，Y值增加時會往下移。進一步，螢幕座標值都是正的，而在修正過的座標值可能是正的或負的，因為基本上它們以自己的座標系為準，X軸（水平）和Y軸（垂直）相交於所選擇的中心點。不像正常座標系，在X軸上方的Y值是負的，在下方是正的。各位必須把這點牢記在心，因為在底下

我們所處理的都是修正過的X，Y值。

第二件工作——找一個方法把圖形送回螢幕，送回的樣子和原來打入的不同——比“只要重新定位”還要難。然而所有的方向均可以利用圖形的反轉，旋轉做到。

旋轉的動作是將每一點以中心點爲準，順着時鐘方向移一段距離（依旋轉的角度而定），然而反轉的動作是取二維的影像並把它翻轉過去，就好像在旋轉撲克牌。明顯的，反轉時需要知道要轉那一軸。

定義旋轉與反轉圖形的方法是一個很有趣的習題（事實上是三個習題與三個方法）。我相信在座標系統運算中一定有此種動作的特定規則存在。我決定利用錯誤嚐試法（trial and error）找出這些規則。拿一張紙和筆，我先定義一個中心點，X軸和Y軸，然後對各種不同組的座標值檢查看看它們在反轉與旋轉時會發生什麼事。對於每一組座標值，我發現的第一件事是，旋轉和反轉只牽涉到二個動作：交換沒有正負號的X和Y值，或是交換一個值或二個值的正負號。這二種動作的組合使用便可產生所有可能的方位。

反轉的動作是最簡單的。我在紙上畫可以知道，不管那一軸做反轉，每一點對於每一軸的距離永遠相同。X和Y值的大小也要相同。正負號卻不一定。例如，對Y軸做反轉，將右上方的點（＋X，－Y）送到左上方（－X，－Y），再將右下方的點（＋X，＋Y）送到左下方（－X，＋Y）。明顯的，對Y軸反轉只要改變X值的正負號即可。同樣的，對X軸反轉只要改變Y值的正負號即可。此步驟寫在程式

行號 1070-1110 與 350-400 。我也分析過是否需要有多次反轉動作的功能，即對某軸反轉二次，或每一軸一次。後來我覺得這不必要，但可以讓讀者當作習題證明看看爲什麼不需要。

在旋轉時比較困難，而做 90°，180°，和 270° 的施轉均要能做到。最簡單的是旋轉 180° 。就像在反轉的情形一樣，某一點旋轉 180° 之後，此點對每一軸的距離還是一樣，因此 X 和 Y 值的大小不變。而正負符號和反轉情形有一點不同。因爲右上方的點（＋X，－Y）要移到左下方（－X，＋Y），右下方的點（＋X，＋Y）要移到左上方（－X，－Y），反之亦是，所以 X 和 Y 值的正負號都要改變。因此旋轉 180° 時，只要把 X 和 Y 值乘上－1 即可。這部份寫在程式行號 1030-1060 和 320-340 。

旋轉 90° 比較不簡單。以時鐘爲例，9 點到 3 點之連線當作 X 軸，12 點到 6 點的連線當作 Y 軸。將時鐘旋轉 90° 後，1 點會轉到 4 點的位置。X 和 Y 值的大小都會改變。在旋轉後，原來一點對 Y 軸的距離變成對 X 軸的距離，反之亦是。所以在旋轉 90° 之後，X 和 Y 值的大小要互相交換。而它們的正負號並沒有明確的關係。然而，確實有一點點關係存在。我依序的測試四個象限，並仔細的觀察 X 和 Y 值的正負號變化。開始將右上方的點（＋X，－Y）移到右下方，可得出（＋X，＋Y）。再旋轉 90°，可得出（－X，＋Y），再旋轉 90° 可得出（－X，－Y）。讀到這裡可發現，在原來象限中 X 值的正負號就是新象限中 Y 值的正負號。因

爲X值的大小變成是Y值的大小，所以我們可以把要旋轉帶正負號的X值都指定給Y值。你也可推知新的X值就是原來的Y值的相反值。所以要得到新的X值，要把原來帶正負號的Y值乘上－1。這二個步驟便可以完成旋轉90°的動作，寫在程式中的行號1030-1060與270-310。爲了使程式儘量短，旋轉270°可利用旋轉90°和180°的副常規來做。

存入磁碟時是將X和Y陣列的值存入一個順序檔案，每個圖形都屬於不同的檔案。雖然有一點浪費磁碟空間，但可以避免使用複雜的檔案管理常規，並允許改變名稱與顯示檔案目錄。最後步驟是在繪圖順序中加入測試動作，假如所選擇的中心點使得圖形畫超出螢幕的範圍，會使得程式停止執行，所以必須要再從頭開始執行。原來的中心點並沒有存在修正過的座標值中，所以每次選擇中心點都要檢查。

此程式是在APPLE II的48K系統上執行，利用DOS來儲存和取出圖形，設立此程式的指令在任何大小的RAM都是一樣的。我相信此程式也可以轉換使用磁帶來儲存，但在本文並不討論（編者註：請看在硬體節中Robert Stein的“卡帶作業系統”）。此機器碼從$800（2048）開始執行，程式並且已經修改過將二部份載入成一個單位，執行時可以重新定機器語言部份的位置。

程式本身是完全自動的，且附有提示，因此我只有一點提示。第一，圖形最好畫在有編號的圖畫紙上，並且圖形的邊最好和螢幕配合，以便檢查螢幕的輸出是否正確。第二，你所要選定的圖形中心點不會自動設立，它可以是螢幕上的

任何一點。你必須記住，所有圖形的旋轉與反轉動作都是根據此中心點，所以選擇中心點時要特別小心。第三，假如圖形非常大，以致於中心點的選擇要非常精確，否則會畫出螢幕外，我們可以把此中心點座標包括在圖形名稱內，以便以後召用此圖形時有所幫助看到圖形名稱便可知道中心點要選在那裡。最後，由於Mr. Suitor 的程式有些限制，有些圖形在碰到邊時不會正常的動作。例如滑行物體，在碰到邊時會變成方塊，而不是繼續移出螢幕外。

為了好玩，在下面找到了一個圖形的座標。稱此圖形為FULSAR SEED，且選擇中心點為 19, 19。當你執行它時，結果一定會令你意外。不管如何，祝愉快！

(*x*,*y*); (10,8); (9,9); (11,9); (9,10); (11,10); (9,11); (10,11); (11,11); (9,1
(11,12); (9,13); (11,13); (10,14); (99,99).

```
0800    ;*************************
0800    ;*                       *
0800    ;*   APPLE LIFESAVER     *
0800    ;* GREGORY L. TIBBETTS   *
0800    ;*                       *
0800    ;*       LIFESAVER       *
0800    ;*                       *
0800    ;*  COPYRIGHT (C) 1981   *
0800    ;*   MICRO INK, INC.     *
0800    ;* CHELMSFORD, MA 01824  *
0800    ;*  ALL RIGHTS RESERVED  *
0800    ;*                       *
0800    ;*************************
0800    ;
0800    ;
0800    ;
```

```
0800 A505          LBLI   LDA $0005
0802 8503                 STA $0003
0804 A504                 LDA $0004
0806 8502                 STA $0002
0808 18                   CLC
0809 6980                 ADC #$80
080B 8504                 STA $0004
080D A505                 LDA $0005
080F 6900                 ADC #$00
0811 C908                 CMP #$08
0813 D00C                 BNE LBLA
0815 A504                 LDA $0004
0817 6927                 ADC #$27
0819 C952                 CMP #$52
081B 1008                 BPL LBLB
081D 8504                 STA $0004
081F A904                 LDA #$0004
0821 8505          LBLA   STA $0005
0823 18                   CLC
0824 60            LBLR   RTS
0825 38            LBLB   SEC
0826 B0FC                 BCS LBLR
0828 20CA08               JSR LBLS
082B 200008        LBLX   JSR LBLI
082E 9001                 BCC LBLC
0830 60                   RTS
0831 A027          LBLC   LDY #$27
0833 98                   TYA
0834 AA                   TAX
0835 A900          LBLH   LDA #$00
0837 994009               STA $0940,Y
083A 997009               STA $0970,Y
083D B102                 LDA ($02),Y
083F F00F                 BEQ LBLE
0841 100A                 BPL LBLD
0843 FE4009               INC $0940,X
0846 FE7009               INC $0970,X
0849 2908                 AND #$08
084B F003                 BEQ LBLE
084D FE4009        LBLD   INC $0940,X
0850 B104          LBLE   LDA ($04),Y
0852 F00F                 BEQ LBLG
0854 1003                 BPL LBLF
0856 FE7009               INC $0970,X
0859 2908          LBLF   AND #$08
085B F006                 BEQ LBLG
085D FE7009               INC $0970,X
0860 FE4009               INC $0940,X
0863 88            LBLG   DEY
0864 CA                   DEX
0865 10CE                 BPL LBLH
0867 A026                 LDY #$26
0869 18                   CLC
086A AD6709               LDA $0967
086D 6D6609               ADC $0966
0870 8506                 STA $0006
0872 AD9709               LDA $0997
0875 6D9609               ADC $0996
0878 8507                 STA $0007
087A 18            LBLW   CLC
087B A506                 LDA $0006
087D 793F09               ADC $093F,Y
```

```
0880 38          SEC
0881 F94209      SBC $0942,Y
0884 8506        STA $0006
0886 C903        CMP #$03
0888 F00E        BEQ LBLK
088A 9004        BCC LBLJ
088C C904        CMP #$04
088E F00E        BEQ LBLL
0890 B102  LBLJ  LDA ($02),Y
0892 F00A        BEQ LBLL
0894 2985        AND #$85
0896 5004        BVC LBLM
0898 B102  LBLK  LDA ($02),Y
089A 0930        ORA #$30
089C B102  LBLM  LDA ($02),Y
089E 18    LBLL  CLC
089F A507        LDA $0007
08A1 796F09      ADC $096F,Y
08A4 38          SEC
08A5 F97209      SBC $0972,Y
08A8 8507        STA $0007
08AA C903        CMP #$03
08AC F00E        BEQ LBLP
08AE 9004        BCC LBLN
08B0 C904        CMP #$04
08B2 F00E        BEQ LBLT
08B4 B104  LBLN  LDA ($04),Y
08B6 F00A        BEQ LBLT
08B8 29F8        AND #$F8
08BA 5004        BVC LBLV
08BC B104  LBLP  LDA ($04),Y
08BE 0903        ORA #$03
08C0 9104  LBLV  STA ($04),Y
08C2 88    LBLT  DEY
08C3 F002        BEQ LBLU
08C5 10B3        BPL LBLW
08C7 4C2B08 LBLU JMP LBLX
08CA A904  LBLS  LDA #$04
08CC 8505        STA $0005
08CE A900        LDA #$00
08D0 8504        STA $0004
08D2 8D6809      STA $0968
08D5 8D8809      STA $0988
08D8 60          RTS
08D9 20CA08      JSR LBLS
08DC 200008 LABD JSR LBLI
08DF 9001        BCC LBLY
08E1 60          RTS
08E2 A027  LBLY  LDY #$27
08E4 B102  LBLC  LDA ($02),Y
08E6 F00A        BEQ LBLZ
08E8 297F        AND #$7F
08EA C910        CMP #$10
08EC 3002        BMI LABA
08EE 0980        ORA #$80
08F0 9102  LABA  STA ($02),Y
08F2 B104  LBLZ  LDA ($04),Y
08F4 F00A        BEQ LABB
08F6 29F7        AND #$F7
08F8 6A          ROR
08F9 9002        BCC LABC
08FB 0904        ORA #$04
08FD 2A    LABC  ROL
```

```
08FE 9104     142           STA ($04),Y
0900 88       143  LABB     DEY
0901 F0D9     144           BEQ LABD
0903 10DF     145           BPL LBLC
             146           END

          *************************
          *                       *
          * SYMBOL TABLE -- V 1.5 *
          *                       *
          *************************

 LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

 ** ZERO PAGE VARIABLES:


 ** ABSOLUTE VARABLES/LABELS

 LBLI   0800  LBLA   0821  LBLR   0824  LBLB   0825  LBLX   082B  LBLC   08
 LBLH   0835  LBLD   084D  LBLE   0850  LBLF   0859  LBLG   0863  LBLW   08
 LBLJ   0890  LBLK   0898  LBLM   089C  LBLL   089E  LBLN   08B4  LBLP   08
 LBLV   08C0  LBLT   08C2  LBLU   08C7  LBLS   08CA  LABD   08DC  LBLY   08
 LBLO   08E4  LABA   08F0  LBLZ   08F2  LABC   08FD  LABB   0900

 SYMBOL TABLE STARTING ADDRESS:6000
 SYMBOL TABLE LENGTH:00FA
```

```
  1 REM   *************************
  2 REM   *                       *
  3 REM   *   APPLE LIFE-SAVER    *
  4 REM   *   GREGORY TIBBETTS    *
  5 REM   *                       *
  6 REM   *       LIFESAVER       *
  7 REM   *                       *
  8 REM   *  COPYRIGHT (C) 1981   *
  9 REM   *    MICRO INK, INC.    *
 10 REM   * CHELMSFORD, MA 01824  *
 11 REM   *  ALL RIGHTS RESERVED  *
 12 REM   *                       *
 13 REM   *************************
 14 REM
 15 LOMEM:2500
 16 DIM HEX$(30)
 30 PRINT "BLOAD LIFE"
 50 GOTO 800
 60 POKE -16302,0: COLOR=0: FOR K=40 TO 47
 70 HLIN 0,39 AT K: NEXT K
 80 KX= PDL (0)-10: IF KX>240 THEN KX=KX1: IF KX<0 THEN KX=0
 90 K1=KX*6:K2=KX*2:K3=500/(K1+50)+1
100 FOR I=1 TO K3
110 CALL GEN
120 FOR K=1 TO K2: NEXT K
```

```
130 CALL MOP
140 FOR K=1 TO SIZE: COLOR=11
150 NEXT I
160 GOTO 80
170 FOR I=1 TO SIZE: COLOR=11
180 X=XCTR+X(I):Y=YCTR+Y(I)
190 IF X<0 OR X>39 OR Y<0 OR Y>39 THEN 1210
200 PLOT X,Y: NEXT I
210 RETURN
220 FOR I=I1 TO I2: FOR J=J1 TO J2
230 COLOR=11: IF RND (L) THEN COLOR=0
240 PLOT I,J
250 NEXT J: NEXT I
260 GOTO 60
270 FOR I=1 TO SIZE
280 X=Y(I):Y=X(I)
290 IF Y(I) THEN X=X*-1
300 X(I)=X:Y(I)=Y
310 NEXT I: RETURN
320 FOR I=1 TO SIZE
330 X(I)=X(I)*-1:Y(I)=Y(I)*-1
340 NEXT I: RETURN
350 FOR I=1 TO SIZE
360 IF XAX THEN 380
370 X=X(I):Y=Y(I)*-1: GOTO 390
380 Y=Y(I):X=X(I)*-1
390 X(I)=X:Y(I)=Y: NEXT I
400 RETURN
410 PRINT D$;"OPEN";A$
420 PRINT D$;"READ";A$
430 FOR I=1 TO 255
440 INPUT X(I),Y(I)
450 IF X(I)=99 OR Y(I)=99 THEN 470
460 NEXT I
470 SIZE=I-1
480 PRINT D$;"CLOSE";A$
490 IF ROT THEN GOSUB 270
500 IF HALF THEN GOSUB 320
510 IF REV THEN GOSUB 350
520 GOSUB 170
530 HALF=0:ROT=0:REV=0:XAX=0:SIZE=0
540 RETURN
550 PRINT D$;"OPEN";A$
560 PRINT D$;"DELETE";A$
570 PRINT D$;"OPEN";A$
580 PRINT D$;"WRITE";A$
590 FOR I=1 TO SIZE
600 PRINT X(I)
610 PRINT Y(I)
620 NEXT I
630 PRINT D$;"CLOSE";A$
640 RETURN
650 FOR I=1 TO 255
660 INPUT X,Y
670 IF X=99 OR Y=99 THEN 720
680 IF X<0 OR X>39 OR Y<0 OR Y>39 THEN 700
690 X(I)=X-XCTR:Y(I)=Y-YCTR: GOTO 710
700 PRINT "INPUT X,Y",X,Y
710 NEXT I
720 X(I)=99:Y(I)=99
730 SIZE=I
```

```
740 RETURN
750 INPUT "INPUT X,Y",X,Y
760 IF X=99 OR Y=99 THEN 60
770 IF X<0 OR X>39 OR Y<0 OR Y>39 THEN 790
780 COLOR=11: PLOT X,Y: GOTO 750
790 PRINT "OUT OF RANGE!": GOTO 750
800 TEXT
810 DIM X(255),Y(255),A$(50),B$(2),D$(1)
820 GEN=2088:MOP=2265:K1=1:K2=1:D$="": REM    D$=CNTRL D
830 CALL -936: VTAB 5: TAB 9: PRINT "CONWAY'S GAME OF LIFE": FOR I=1 TO
    700: NEXT I
840 GR
850 PRINT "DO YOU WISH TO: 1.PLAY OR 2.CREATE"
860 INPUT "A NEW PATTERN FILE (1/2).",C1
870 IF C1=2 THEN 1140
880 INPUT "SPEED=PDL(0):SET DEFAULT (0-255)",KX1
890 PRINT "DO YOU WISH: 1.RANDOM PATTERN 2.PATTERN"
900 INPUT "FROM DISK OR 3.STANDARD: (1/2/3)",C1
910 IF C1=3 THEN 990
920 IF C1=2 THEN 1010
930 INPUT "X DIRECTION LIMITS ",I1,I2
940 IF I1<0 OR I2>39 OR I1>I2 THEN 930
950 INPUT "Y DIRECTION LIMITS ",J1,J2
960 IF J1<0 OR J2>39 OR J1>J2 THEN 950
970 INPUT "ONE IN 'N' CELLS WILL LIVE: ENTER N",L
980 GOTO 220
990 PRINT "ENTER YOUR PATTERN (X,Y):99,99 EXITS"
1000 GOTO 750
1010 INPUT "WHAT FIGURE NAME",A$
1020 INPUT "ENTER CENTER COORD'S (X,Y)",XCTR,YCTR
1030 INPUT "ENTER ROTATION (0/90/180/270)",ROT
1040 IF ROT=180 OR ROT=270 THEN HALF=1
1050 IF ROT=90 OR ROT=270 THEN ROT=1
1060 IF ROT<>1 THEN ROT=0
1070 INPUT "ENTER 1.REVERSED OR 2.STANDARD (1/2)",REV
1080 IF REV>1 THEN REV=0: IF NOT REV THEN 1110
1090 INPUT "REVERSE ON 1.X-AXIS OR 2.Y-AXIS (1/2)",XAX
1100 IF XAX>1 THEN XAX=0
1110 GOSUB 410
1120 INPUT "ANOTHER FIGURE (Y/N)",B$: IF B$="N" THEN 60
1130 PRINT "CAUTION: FIGURES MAY OVERWRITE!": GOTO 1010
1140 INPUT "ENTER CENTER COORD'S (X,Y)",XCTR,YCTR
1150 PRINT "ENTER ALL LIVE CELLS (X,Y):99,99 EXITS"
1160 GOSUB 650
1170 INPUT "ENTER NAME FOR THIS FIGURE",A$
1180 GOSUB 550
1190 PRINT "TESTING": GOSUB 410
1200 GOTO 60
1210 PRINT "PLOT ABORTED/FIGURE WENT OFF SCREEN"
1220 PRINT "MOVE CENTERPOINT:X AND Y WHEN ABORTED"
1230 PRINT "WERE ";X;",";Y: POP : POP
1240 IF I=1 THEN 1020:IE=I-1: COLOR=0: FOR I=1 TO IE
1250 PLOT X(I)+XCTR,Y(I)+YCTR: NEXT I: GOTO 1020
1260 REM   ADAPTATION BY GREG TIBBETTS OF RICHARD SUITOR'S PROGRAM IN
1265 REM   "BEST OF MICRO" VOLUME II 1979
1270 REM   LINES 0-50 PROGRAM SET-UP
1280 REM   60-160 SPEED AND GENERATION
1290 REM   170-210 GENERAL PLOT SUBR.
1300 REM   220-260 RANDOM PLOT SUBR.
1310 REM   270-340 ROTATION SUBR'S.
1320 REM   350-400 REVERSAL SUBR.
1330 REM   410-540 DISK READ SUBR.
1340 REM   550-640 DISK WRITE SUBR.
```

```
 1345 REM   650-740 DISK INPUT SUBR.
 1350 REM   750-790 STANDARD INPUT SUBR.
 1360 REM   800-840 INITIALIZATION
 1370 REM   850-920 MODE SELECTION
 1380 REM   930-1200 USER INPUT/SELECT
 1390 REM  1210-1250 PLOT ABORT SUBR.
10000 END
```

APPLAYER 音樂解析器

作者：Richard F. Suitor

APPLE 能產生聲音的能力是衆所皆知的。但一些APPLE 的使用者都很少用，乃是因爲在程式中很難產生好聽的音調。此 Applayer音樂解析器將解決此問題，並且提供一個很直接的方法使你的APPLE 能產生眞正的音樂。

此程式不只是一個音調產生常規，它是一個音樂解析器。它能使你建立一表格來定義半音（ half-tone ），並且用簡單的碼便能使聲音持續。它能使你很容易的達到你的目的。

此程式是用6502 的組合語言寫的，以便能普遍的使用。只要揚聲器的介面類似，則此組合語言程式可用於任何系統上。處理一個特殊位址（ C030 ）改變通過APPLE 揚聲器的電流，從開（ ON ）到關（ OFF ），或從關到開；動作就好像在按鈕一樣。因此程式週期性的利用LDA C030 處理此位址便可達到發聲的效果。任何介面均可容易的使用類似的指令來做。一個不同的軟體介面可能需要一些修改。

音調的產生是利用一個時間迴路（ timing loop ）來

計算定時週期（ clock cycle ）的次數，N。音調的變化大約從 20Hz 到 8KHz

Applayer解析位元組以產生不同的效果。在此有二種位元組型式：

音調位元組——第 7 個位元不設爲 1 。

控制位元組——設立第 7 個位元爲 1 。

一個音調位元組允許你在 16 個半音中選一個，在連音（ duration ）方面可以從 1 個到 8 個八分之一音符。位元 0 ，1 ，2 ，3 是控制半音的選擇；位元 4 ，5 ，6 是控制連音部份。

位元	7 6 5 4 3 2 1 0
音調位元組	0（連音）（半音）

控制位元組使你能改變音樂的速度，16 個半音的音調範圍，休止符，音調的波長，以及從表格的中間跳到另一個表格。

控制位元組表

十六進位	十進位	功能
81	129	下三個位元組是新的波形格式。
82	130	JMP ——接着是新表格的位址。
83	131	JSR ——接着是新表格的位址。
9N	144 + N	表示需休止N個十六分之一個音符。
AN	160 + N < 32	選擇音調的範圍。計分成4個八度音階。
CN	192 + N < 32	控制速度。音符的長度和N成正比，最長大約3.5秒。
FF	255	RETURN 。停止解析動作。如83 JSR 指令的返回動作，返回到Applayer 。

對散頁樂譜（sheet music ）使用Applayer ，首先要決定半音的範圍。這在曲子的中間有時是一定要改變的。例如，“ Turkey in the Straw ”曲子，等一下在後面會提到，是C調；在曲子的前半部我使用下表：

音符　C D E F G A B C D

音調＃0 2 4 5 7 9 B C E

可利用一個控制位元組設立此音調範圍，B0。在合唱方面，旋律的範圍往上移；音調的範圍設爲B7，表格如下：

音符　G A B C D E F G A

音調＃0 2 4 5 7 9 A C E

（眞正的曲調是由波形格式決定的，而不僅是音調範圍控制位元組決定。若波形格式爲 050505，C調的基音大約是346 Hz，很接近F調）。

休止符可利用一個 9N控制位元組和一個音調位元組。例如，94 10表示四分之一休止符，98 30表示二分之一休止符，等等。平常若要將音調清楚的分開可用 91，或是要連續時可用 90。

讓我們來試着建立一個表格，以便使APPLE 能奏出一曲旋律。我們選擇“Twinkle, Twinkle Little Star”開始。此調子計有 4 行；第 1 行和第 4 行一樣，第 2 行和第 3 行一樣。表格設計如下：

1. 設立音調的範圍，音調形式與速度。
2. JSR到第 1 行的表格。
3. JSR到第 2 行的表格。
4. 重複＃3。

5. 重複＃2。
6. 返回（Return）。
7. 第1行的表格與返回。
8. 第2行的表格與返回。

因爲Applayer並沒有符號化，所以在建表格時要以相反方向來建，所以在步驟2-6中才知道要跳到那裡（即跳到步驟7與步驟8）。第1行的表格可建在0B00，如下：

```
0B00- 10 10 17 17 19 19 37 15
0B08- 15 14 14 12 12 30 FF FF
```

第2行的表格可建在0B10，如下：

```
0B10- 17 17 15 15 14 14 32 FF
```

現在我們可從步驟1開始。我建議如下；你可能想要做一些更改也無妨：

```
0B20- B0 81 05 05 05 E0 91
```

上述決定了音調的範圍，波長形式，速度，以及一個使各個音符分開的十六分之一休止符。若不分開，可用 90 而不必用 91 。步驟2-6如下：

```
0B20-                      83
0B28- 00 0B 83 10 0B 83 10 0B
0B30- 83 00 0B FF
```

如此便完成了“ Twinkle , Twinkle ”的表格。現在我們必須告訴 Applayer它在那裡，並打開它。從BASIC我們首先要在第0頁中設立一些值，再JSR到Applayer ：（在執行前不要忘了設立LOMEM；此表格可設爲2900。）

```
100   POKE   19,32    （表格位址0B20的低位元組）
110   POKE   20,11    （表格位址0B20的高位元組）
120   POKE    1,8     （Applayer程式第1頁的高位元
                        組）
130   POKE   17,8     （16和17含有音調表格位址）
140   POKE   16,0
150   CALL   23,46    （跳到副常規092A）
```

我們也可利用一個小組合語言程式設立第0頁的一些值。看常規ZERO，在程式的09C0位址。

這些啓始動作可以更簡單，只要保留第A00頁當作“表格的內容”針對記憶體中不同的音調表格。在“ Twinkle, Twinkle ”中，我們加入下面表格：

```
0A20- 82 20 0B
```

它將馬上跳到0B20去。利用此種便利處，我們可以只改變9D0（十進位爲2512）的值便可從一表格移到另一表格。

我們也可從BASIC使用，只要改變最後一個指令爲

RTS：

```
100   POKE   2512,32  （表格的低位元組）
110   POKE   2538,96  （改變09EA的指令爲RTS）
120   CALL   2496     （跳到副常規9C0）
```

從監督系統：＊9D0：20
＊9C0 G

便可執行。

假如你聽膩了“Twinkle, Twinkle”，你可能希望聽一下“Turkey in the Straw”。表格如下，其結構留着當練習題。

從監督系統：＊9D0：0
＊9C0 G

便可執行。

（編者註：磁碟上包括一個APPLAYER的整數BASIC驅動常規，稱爲APPLAYER MENU。此程式會自動載入與執行此音樂解析程式，並允許使用者選擇二首曲子的任何一曲。（這二首曲子已包括在APPLAYER的檔案中）。若系統沒有整數BASIC，則可依照前面所述來做。）

“Turkey in the Straw”的音符表格

0A00:	83	90	0F	83	90	0F	FF	
0F00:	90	1C	1A	92	38	90	18	1A
0F08:	18	13	10	11	91	13	13	33
0F10:	33	90	18	1A	92	3C	3C	90
0F18:	1C	1A	18	1A	91	1C	38	18
0F20:	38	90	1C	1A	92	38	90	18
0F28:	1A	18	13	91	10	11	13	53
0F30:	33	90	18	1A	91	3C	3F	90
0F38:	1F	1C	18	1A	1C	18	92	3A
0F40:	94	78	91	FF				
0F50:	81	55	55	55	FF			
0F58:	81	05	05	05	FF			
0F60:	15	18	18	15	78	FF		
0F68:	16	1A	1A	16	7A	FF		
0F70:	1D	1D	1D	1D	18	18	18	18
0F78:	35	15	15	33	90	11	13	91
0F80:	15	18	18	18	90	18	15	11
0F88:	13	91	15	15	13	13	71	FF
0F90:	83	58	0F	D4	B0	83	50	0F
0F98:	B7	83	60	0F	83	50	0F	83
0FA0:	60	0F	83	50	0F	83	68	0F
0FA8:	83	50	0F	83	68	0F	83	50
0FB0:	0F	83	70	0F	FF			

音 調 表 格

0800:	A0	03	68	03	38	03	08	03
0808:	E0	02	B8	02	90	02	68	02
0810:	48	02	28	02	08	02	E8	[illegible]
0818:	D0	01	B4	01	9C	01	84	01
0820:	70	01	5C	01	48	01	34	01
0828:	24	01	14	01	04	01	F4	00
0830:	E8	00	DA	00	CE	00	C2	00
0838:	B8	00	AE	00	A4	00	9A	00
0840:	92	00	8A	00	82	00	7A	00
0848:	74	00	6D	00	67	00	61	00
0850:	5C	00	57	00	52	00	4D	00
0858:	49	00	45	00	41	00	3D	00

```
0800              ;*************************
0800              ;*                       *
0800              ;*    APPLAYER MUSIC     *
0800              ;*     INTERPRETER       *
0800              ;*  RICHARD F. SUITOR    *
0800              ;*                       *
0800              ;*      APPLAYER         *
0800              ;*                       *
0800              ;*  COPYRIGHT (C) 1981   *
0800              ;*    MICRO INK, INC.    *
0800              ;* CHELMSFORD, MA 01824  *
0800              ;*  ALL RIGHTS RESERVED  *
0800              ;*                       *
0800              ;*************************
0800              ;
0800              ;
0800              ;
0800              ;
0860                      ORG $0860
0860                      OBJ $0860
0860              ;
0860              ;
0860 EA           TIME    NOP
0861 EA                   NOP
0862 EA                   NOP
0863 88           TIMEA   DEY
0864 8545                 STA $0045     ;ANY INNOCUOUS 3 CYCLE INSTRUCTION
0866 D0FB                 BNE TIMEA     ;BASIC 8 CYCLE LOOP
0868 F005                 BEQ TIMEC
086A 88           TIMEB   DEY
086B EA                   NOP
086C EA                   NOP
086D D0F4                 BNE TIMEA
086F 2404         TIMEC   BIT $0004     ;START CHECK OF BIT PATTERN
0871 38                   SEC           ;IN 2, 3, AND 4
0872 3002                 BMI TIMED
0874 EA                   NOP
0875 18                   CLC
0876 2603         TIMED   ROL $0003
0878 2602                 ROL $0002
087A 2604                 ROL $0004
087C 9003                 BCC TIMEE
087E AD30C0               LDA $C030     ;TOGGLE SPEAKER
0881 C606         TIMEE   DEC $0006     ;DURATION OF NOTE IN
0883 D005                 BNE TIMEF     ;NO. OF CYCLES IN LOCATIONS
0885 C607                 DEC $0007     ;6 AND 7
0887 D005                 BNE TIMEG
0889 60                   RTS
088A EA           TIMEF   NOP           ;TIMING EQUALIZATION
088B EA                   NOP
088C D000                 BNE TIMEG
088E A405         TIMEG   LDY $0005
0890 6C0000               JMP ($0000)
0893              ;
0893              ;SCALING ROUTINE FOR CYCLE DURATION
0893              ;CALCULATION LOC 6,7 = A REG *LOC
0893              ;50, 51
0893              ;
```

```
0893 8545    SCALE  STA $0045
0895 A900           LDA #$C0
0897 8506           STA $0006
0899 8507           STA $0007
089B A205           LDX #$05
089D 18             CLC
089E 6607    SCALEX ROR $0007
08A0 6606           ROR $0006
08A2 4645           LSR $0045
08A4 900C           BCC SCALEA
08A6 A506           LDA $0006
08A8 6550           ADC $0050
08AA 8506           STA $0006
08AC A507           LDA $0007
08AE 6551           ADC $0051
08B0 8507           STA $0007
08B2 CA      SCALEA DEX
08B3 10E9           BPL SCALEX
08B5 E607           INC $0007      ; SIMPLE LOGIC IN TIMING ROUTIN
08B7 60             RTS
08BE                ORG $08BE
08BE         ;
08BE         ;NOTE PLAYING ROUTINE Y REG
08BE         ;HAS HALF-TONE INDEX
08BE         ;
08BE A512    NOTE   LDA $0012      ;NOTE LENGTH
08C0 8552           STA $0052
08C2 A50F           LDA $000F      ;NOTE TABLE OFFSET
08C4 8510           STA $0010
08C6 B110           LDA ($0010),Y ;LOW ORDER BYTE OF
08C8 38             SEC            ;MACHINE CYCLES PER PERIOD
08C9 8554           STA $0054
08CB E935           SBC #$35       ;CYCLES USED UP TIMING OVERHEAD
08CD 8508           STA $0008
08CF C8             INY
08D0 B110           LDA ($0010),Y ;HIGH ORDER BYTE  OF MACHINE
08D2 8555           STA $0055      ;CYCLES PER PERIOD
08D4 E900           SBC #$00
08D6 8509           STA $0009
08D8 A900           LDA #$00
08DA 8550           STA $0050
08DC 8551           STA $0051
08DE 8553           STA $0053
08E0 A010           LDY #$10
08E2 202403         JSR $0324
08E5         ;
08E5         ;
08E5         ;THE ROUTINE AT $324 EMULATES THE OLD
08E5         ;MONITOR DIVIDE ROUTINE, WHICH DIVIDES
08E5         ;LOCS 54,55 BY 52,53 AND LEAVES THE
08E5         ;RESULT IN 50,51 FOR THE SCALING
08E5         ;ROUTINE. THIS DIVIDE ROUTINE IS LISTED
08E5         ;IN THE REFERENCE MANUAL ON P.162 ($FB81)
08E5         ;
08E5 A508           LDA $0008
08E7 48             PHA
08E8 4609           LSR $0009
```

```
08EA 6A                     ROR
08EB 4609                   LSR $0009
08ED 6A                     ROR
08EE 4609                   LSR $0009
08F0 6A                     ROR
08F1 8505                   STA $0005      ;NO. OF 8 CYCLE LOOPS
08F3 68                     PLA
08F4 2907                   AND #$07       ;LEFT OVER CYCLES DETERMINE
08F6 AA                     TAX            ;ENTRY POINT
08F7 BDF809                 LDA TTABLE,X   ;TABLE OF ENTRY POINTS
08FA 850C                   STA $0000      ; FOR TIMING LOOP
08FC A50E                   LDA $000E      ;NOTE DURATION, QUARTER,
08FE 38                     SEC            ;HALF
08FF E50D                   SBC $000D      ;REST PART OF NOTE
0901 F00F                   BEQ NOTEB      ;IF NOTHING TO DO
0903 209308                 JSR SCALE      ;SCALING ROUTINE
0906 A202                   LDX #$02       ;START PATTERN LOAD
0908 B50A           NOTEA   LDA $0A,X
090A 95C2                   STA $C2,X
090C CA                     DEX
090D 10F9                   BPL NOTEA
090F 206FC8                 JSR TIMEC      ;TIMING ROUTINE
0912 A50D           NOTEB   LDA $000D      ;REST PART OF NOTE
0914 F00E                   BEQ MAIN       ;IF NOTHING TO DO
0916 209308                 JSR SCALE      ;SCALING ROUTINE
0919 A900                   LDA #$00
091B 8502                   STA $0002      ;ZERO OUT PATTERN FOR
091D 8503                   STA $0003      ;REST PART
091F 8504                   STA $0004
0921 206F08                 JSR TIMEC      ;TIMING
0924                        ORG $0924
0924                ;
0924                ;MAIN PART OF INTERPRETER
0924                ;ENTRY AT "ENTRY"
0924                ;
0924 E613           MAIN    INC $0013      ;TABLE ADDRESS
0926 D002                   BNE ENTRY
0928 E614                   INC $0014
092A A000           ENTRY   LDY #$00
092C B113                   LDA ($0013),Y  ;NEXT TABLE BYTE
092E 3012                   BMI MAINA      ;TO CONTROL SECTION
0930 48                     PHA
0931 290F                   AND #$0F       ;TONE
0933 0A                     ASL
0934 A8                     TAY
0935 68                     PLA
0936 2970                   AND #$70       ;DURATION
0938 4A                     LSR
0939 4A                     LSR
093A 4A                     LSR
093B 6902                   ADC #$02       ;TOTAL DURATION IN 16THS
093D 850E                   STA $000E
093F 4CBE08                 JMP NOTE       ;PLAY NOTE
0942 C9FD           MAINA   CMP #$FD       ;C0 + 3D IS LONGEST NOTE FOR
0944 9001                   BCC MAINB      ;SCALING REASONS
0946 60                     RTS
0947 48             MAINB   PHA
0948 0A                     ASL
```

```
0949 1007             BPL MAINC
094B 68               PLA
094C 293F             AND #$3F          ;NOTE LENGTH
094E 8512             STA $0012
0950 B0D2             BCS MAIN          ;UNCONDITIONAL BRANCH
0952 0A         MAINC ASL
0953 1008             BPL MAIND
0955 68               PLA
0956 291F             AND #$1F          ;TONAL RANGE INDEX
0958 0A               ASL
0959 850F             STA $000F
095B 90C7             BCC MAIN          ;UNCONDITIONAL BRANCH
095D 0A         MAIND ASL
095E 1007             BPL MAINE
0960 68               PLA
0961 290F             AND #$0F          ;REST FRACTION
0963 850D             STA $000D
0965 90BD             BCC MAIN          ;UNCONDITIONAL BRANCH
0967 0A         MAINE ASL
0968 1003             BPL MAING
096A 68         MAINF PLA
096B 90B7             BCC MAIN          ;DUMMY, CONTROLS NOT INTERPRETED
096D 0A         MAING ASL
096E 30FA             BMI MAINF
0970 0A               ASL
0971 102B             BPL MAINI
0973 68               PLA
0974 AA               TAX               ;JSR AND JMP SECTION
0975 4A               LSR
0976 900A             BCC MAINH
0978 A513             LDA $0013         ;JSR SECTION, PUSH RETURN TABLE
097A 6901             ADC #$01          ;ADDRESS ON TO STACK
097C 48               PHA
097D A514             LDA $0014
097F 6900             ADC #$00
0981 48               PHA
0982 C8         MAINH INY
0983 B113             LDA ($0013),Y     ;GET NEW ADDRESS
0985 48               PHA
0986 C8               INY
0987 B113             LDA ($0013),Y
0989 8514             STA $0014
098B 68               PLA
098C 8513             STA $0013
098E 8A               TXA               ;AND STORE IT FROM BEGINNING
098F 4A               LSR               ;OF SELECTION
0990 9098             BCC ENTRY         ;JMP
0992 202A09           JSR ENTRY         ;JSR
0995 68               PLA
0996 8514             STA $0014         ;PULL ADDRESS AND STORE IT
0998 68               PLA
0999 8513             STA $0013
099B 18               CLC
099C 9086             BCC MAIN          ;UNCONDITIONAL BRANCH
099E 68         MAINI PLA
099F A003             LDY #$03          ;GET NEW PATTERN AND
09A1 B113       MAINJ LDA ($0013),Y     ;STORE IT
09A3 990900           STA $0009,Y
```

```
09A6 88                      DEY
09A7 D0F8                    BNE MAINJ
09A9 A513                    LDA $0013
09AB 6903                    ADC #$03          ;JUMP OVER PATTERN
09AD 8513                    STA $0013
09AF 9002                    BCC MAINK
09B1 E614                    INC $0014
09B3 4C2409         MAINK    JMP MAIN
09C0                         ORG $09C0
09C0                ;
09C0                ;INITIALIZATION FOR ZERO PAGE
09C0                ;
09C0 D8             ZERO     CLD               ;JUST IN CASE
09C1 A900                    LDA #$00
09C3 8510                    STA $0010
09C5 A908                    LDA #$08
09C7 8511                    STA $0011
09C9 8501                    STA $0001
09CB A90A                    LDA #$0A
09CD 8514                    STA $0014         ;NOTE TABLE PAGE
09CF A920                    LDA #$20
09D1 8513                    STA $0013         ;NOTE TABLE BYTE
09D3 A901                    LDA #$01
09D5 850D                    STA $000D         ;REST 16THS
09D7 A920                    LDA #$20
09D9 8512                    STA $0012         ;NOTE LENGTH, CONTROLS TEMPO
09DB A920                    LDA #$20
09DD 850F                    STA $000F         ;TONAL RANGE INDEX
09DF A905                    LDA #$05
09E1 850A                    STA $000A         ;WAVE SHAPE PATTERN
09E3 850B                    STA $000B
09E5 850C                    STA $000C
09E7 202A09                  JSR ENTRY         ;TO APPLAYER
09EA 4C69FF                  JMP $FF69         ;TO MONITOR, AFTER THE BEEP
09F8                         ORG $09F8
09F8                ;
09F8                ;TABLE OF ENTRY POINTS FOR TIMING ROUTINE
09F8                ;
09F8 636A62         TTABLE HEX 636A626D616C606B
09FB 6D616C
09FE 606B
                             END
```

```
**************************
*                        *
* SYMBOL TABLE -- V 1.5  *
*                        *
**************************
```

```
LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:


** ABSOLUTE VARABLES/LABELS
```

TIME	0860	TIMEA	0863	TIMEB	086A	TIMEC	086F	TIMED	0876	TIMEE	0881
TIMEF	088A	TIMEG	088E	SCALE	0893	SCALEX	089E	SCALEA	08B2	NOTE	08BF
NOTEA	0908	NOTEB	0912	MAIN	0924	ENTRY	092A	MAINA	0942	MAINB	0947
MAINC	0952	MAIND	095D	MAINE	0967	MAINF	096A	MAING	096D	MAINH	0982
MAINI	099E	MAINJ	09A1	MAINK	09B3	ZERO	09CC	TTABLE	09E8		

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:00FA

```
  1 REM *************************
  2 REM *                       *
  3 REM *    APPLAYER MUSIC     *
  4 REM *     INTERPRETER       *
  5 REM *                       *
  6 REM *   BY RICHARD SUITOR   *
  7 REM *                       *
  8 REM *     APPLAYER MENU     *
  9 REM *                       *
```


```
 10 REM *  COPYRIGHT (C) 1981   *
 11 REM *    MICRO INK, INC.    *
 12 REM * CHELMSFORD, MA 01824  *
 13 REM *  ALL RIGHTS RESERVED  *
```


```
 14 REM *                       *
 15 REM *************************
 16 REM
 17 REM
 18 PRINT "BLOAD APPLAYER"
 19 LOMEM:4095
 20 START=2496:LOBYTE=2512
 30 IMAX=2
100 CALL -936
110 TAB 13
120 PRINT "APPLAYER MENU"
130 VTAB 4
140 PRINT "1  -  TWINKLE, TWINKLE"
150 PRINT "2  -  TURKEY IN THE STRAW"
195 VTAB 19
200 INPUT "WHICH NUMBER",I
220 IF I<0 OR I>IMAX THEN 100
230 IF I=0 THEN END
240 IF I=1 THEN J=32
250 IF I=2 THEN J=0
300 POKE LOBYTE,J
320 CALL START
350 GOTO 100
```

改進星際戰爭的音響效果

作者： **William M. Shryock, Jr.**

製作科幻電影的音效製作小組可能會問：“星際戰爭所發出的聲音是什麼效果？”，APPLE II 的回答是……。

```
 1 REM   ******************************
 2 REM   * STAR BATTLE SOUND EFFECTS *
 3 REM   *           BY              *
 4 REM   *    WILLIAM SHRYOCK, JR.   *
 5 REM   *    COPYRIGHT (C) 1981     *
 6 REM   *      MICRO INK, INC.      *
 7 REM   *    ALL RIGHTS RESERVED    *
 8 REM   ******************************
10 POKE 0,160: POKE 1,1: POKE 2,162: POKE 3,0: POKE 4,138: POKE 5,24: POKE
   6,233: POKE 7,1: POKE 8,208: POKE 9,252: POKE 10,141
20 POKE 11,48: POKE 12,192: POKE 13,232: POKE 14,224: POKE 15,150: POKE
   16,208: POKE 17,242: POKE 18,136: POKE 19,208: POKE 20,237: POKE 21
   ,96
30 CALL -936: VTAB 12: TAB 9: PRINT "STAR BATTLE SOUND EFFECTS"
40 SHOTS= RND (15)+1
50 LENGTH= RND (11)*10+120
60 POKE 1,SHOTS: POKE 15,LENGTH: CALL 0
70 FOR DELAY=1 TO RND (1000): NEXT DELAY
80 GOTO 40
```

此程式可用於低解析度的程式，而不必重新設立HIMEM。它也可以從BASIC載入。

Galacti-Cube

作者：**Bob Bishop**

你是宇宙戰艦的艦長，負責探測我們宇宙之外的秘密。你發現一個巨大的立方體漂浮在太空中。而你已將戰艦駛入立方體，但現在你卻找不到出路了！怎麼辦？

GALACTI-CUBE 是一個簡單的三維空間遊戲。你是在一個 3 × 3 × 3 的立方體中，你必須在移動 40 步之前找到出路，否則你就輸了。移動時可打入N，S，E，W，U，或D，分別表示往北移，往南移，往東移，往西移，往上移。雖然它看起來很小，但在裡面有 27 個小空間存在，將使你的工作不會很簡單。

此程式完全以APPLE II 的整數BASIC寫的，系統記憶體至少要有 8K 位元組。事實上，因爲程式沒有用到機器語言，圖形，或特殊音效，它可以很容易的轉換給CRT-型式的計算機使用（如PET, TRS-80，等）。

```
 10 REM  *************************
 12 REM  *                       *
 14 REM  *      CALACTI-CUBE     *
 16 REM  *      R.J.  BISHOP     *
 18 REM  *                       *
 20 REM  *  COPYRIGHT (C) 1981   *
 22 REM  *     MICRO INK, INC.   *
 24 REM  * CHELMSFORD, MA 01824  *
 26 REM  *  ALL RIGHTS RESERVED  *
 28 REM  *                       *
 29 REM  *************************
 30 DIM BOX(27),QUE(27),NODE(6),BIT(6),A$(5)
 40 GOSUB 9000
 50 GOSUB 1000
 60 VTAB 23: TAB 5: PRINT "(HIT ANY KEY TO START THE GAME) ";
 70 GOSUB 4000: GOSUB 5000
 90 LOC=14:OLD=LOC:FUEL=40
100 REM MAIN LOOP
110 GOSUB 2000
150 CALL -936: PRINT : PRINT : PRINT "   COMMAND:"
160 PRINT : TAB 7: GOSUB 4000: CALL -936
165 IF A$="" THEN 150
170 IF A$(1,1)#"F" THEN 250
180 CALL -936: PRINT : PRINT "  YOU HAVE ";FUEL
190 PRINT : PRINT "  FUEL UNITS"
210 FOR K=1 TO 1000: NEXT K: GOTO 150
250 Z=(OLD-1)/9+1
260 Y=(((OLD-1)/3) MOD 3)+1
270 X=((OLD-1) MOD 3)+1
300 IF A$="E" THEN X=X+1
310 IF A$="W" THEN X=X-1
320 IF A$="N" THEN Y=Y+1
330 IF A$="S" THEN Y=Y-1
340 IF A$="U" THEN Z=Z+1
350 IF A$="D" THEN Z=Z-1
360 LOC=X+3*(Y-1)+9*(Z-1)
370 IF LOC<>OLD THEN 390
380 PRINT "": GOTO 150
390 IF X<1 OR X>3 OR Y<1 OR Y>3 THEN 700
400 IF BOX(OLD)>=32 AND Z=0 THEN 800
410 VAL=BOX(OLD): IF VAL>=32 THEN VAL=VAL-32
420 IF VAL>=16 AND Z=4 THEN 800
430 IF Z<1 OR Z>3 THEN 700
450 BITS=BOX(OLD)
460 WAY=BITS-2*(BITS/2):BITS=BITS/2
470 IF WAY=0 AND A$="E" THEN 700
480 WAY=BITS-2*(BITS/2):BITS=BITS/2
490 IF WAY=0 AND A$="W" THEN 700
500 WAY=BITS-2*(BITS/2):BITS=BITS/2
505 IF WAY=0 AND A$="N" THEN 700
510 WAY=BITS-2*(BITS/2):BITS=BITS/2
515 IF WAY=0 AND A$="S" THEN 700
520 WAY=BITS-2*(BITS/2):BITS=BITS/2
525 IF WAY=0 AND A$="U" THEN 700
530 WAY=BITS-2*(BITS/2):BITS=BITS/2
535 IF WAY=0 AND A$="D" THEN 700
540 WAY=BITS-2*(BITS/2):BITS=BITS/2
550 FUEL=FUEL-1: IF FUEL>0 THEN 100
```

```
560 CALL -936: PRINT "   YOU ARE"
565 PRINT
570 PRINT "    OUT OF"
575 PRINT
580 PRINT "     FUEL!";
590 GOTO 830
700 CALL -936: PRINT " THAT DIREC-"
710 PRINT : PRINT " TION HAS AN"
720 PRINT : PRINT " OBSTRUCTION";
730 FOR K=1 TO 1000: NEXT K: GOTO 150
800 CALL -936: PRINT "YOU FOUND THE"
810 PRINT : PRINT " EXIT IN ONLY"
820 PRINT : PRINT "  ";41-FUEL;" MOVES!";
830 GOSUB 2700
840 FOR K=1 TO 2500: NEXT K
850 CALL -936: END
900 END
1000 REM GENERATE THE MAZE
1010 FOR K=1 TO 27
1020 BOX(K)=128
1030 NEXT K
1040 BOX(14)=0
1050 QUE(1)=14:QBIG=1
1060 XQBIG=1
1100 FOR K=1 TO QBIG
1110 IND=QUE(K)
1140 KNT=0:ROAD=1:DEL=1
1150 FOR J=0 TO 2
1160 SET=3*DEL
1170 FOR L=0 TO 1
1180 NDX=IND+DEL
1190 IF NDX<1 THEN 1400
1200 IF (NDX-1)/SET<>(IND-1)/SET THEN 1400
1250 IF BOX(NDX)<128 THEN 1400
1300 KNT=KNT+1:NODE(KNT)=NDX:BIT(KNT)=ROAD
1400 DEL=-DEL:ROAD=ROAD+ROAD
1450 NEXT L
1460 DEL=SET
1470 NEXT J
1500 IF KNT=0 THEN 1600
1510 NDX= RND (KNT)+1:XQBIG=XQBIG+1
1520 QUE(XQBIG)=NODE(NDX)
1530 BOX(IND)=BOX(IND)+BIT(NDX)
1540 TIB=2*BIT(NDX)
1550 IF TIB=4 OR TIB=16 OR TIB=64 THEN TIB=TIB/4
1590 BOX(NODE(NDX))=BOX(NODE(NDX))+TIB-128
1600 NEXT K
1610 QBIG=XQBIG: IF QBIG<27 THEN 1100
1700 HOLE=2* RND (2)+6* RND (2)+18* RND (2)+1
1710 OPEN=16: IF HOLE<14 THEN OPEN=32
1720 BOX(HOLE)=BOX(HOLE)+OPEN
1800 RETURN
2000 REM UPDATE THE DISPLAY
2005 GOSUB 2700
2010 Z=(OLD-1)/9+1
2020 Y=(((OLD-1)/3) MOD 3)+1
2030 X=((OLD-1) MOD 3)+1
2040 VTAB 13-Y-Y
```

```
2050 TAB 8*Z+X+X-7
2060 PRINT "-"
2110 Z=(LOC-1)/9+1
2120 Y=(((LOC-1)/3) MOD 3)+1
2130 X=((LOC-1) MOD 3)+1
2140 VTAB 13-Y-Y
2150 TAB 8*Z+X+X-7
2170 POKE PEEK (36)+ PEEK (40)+256* PEEK (41),109
2200 BITS=BOX(LOC)
2210 VT=20:T=34:A$="EAST": GOSUB 2500
2220 VT=22:T=34:A$="WEST": GOSUB 2500
2230 VT=20:T=28:A$="NORTH": GOSUB 2500
2240 VT=22:T=28:A$="SOUTH": GOSUB 2500
2250 VT=20:T=24:A$="UP": GOSUB 2500
2260 VT=22:T=23:A$="DOWN": GOSUB 2500
2300 GOSUB 2600
2400 OLD=LOC
2450 RETURN
2500 WAY=BITS-2*(BITS/2):BITS=BITS/2
2510 MODE=127: IF WAY THEN MODE=255
2520 POKE 50,MODE: VTAB VT: TAB T: PRINT A$: POKE 50,255
2550 RETURN
2600 VTAB 19: TAB 5
2610 POKE 32,2
2630 POKE 33,14
2660 POKE 34,17
2680 POKE 35,22
2690 RETURN
2700 POKE 32,0
2710 POKE 33,40
2720 POKE 34,0
2730 POKE 35,24
2750 RETURN
4000 REM 'GET' FROM THE KEYBOARD
4010 POKE -16368,0
4020 CHAR= PEEK (-16384): IF CHAR<128 THEN 4020
4030 POKE -16368,0:A$="?"
4080 IF CHAR=141 THEN A$=""
4090 IF CHAR=196 THEN A$="D"
4100 IF CHAR=197 THEN A$="E"
4110 IF CHAR=198 THEN A$="F"
4120 IF CHAR=206 THEN A$="N"
4130 IF CHAR=211 THEN A$="S"
4140 IF CHAR=213 THEN A$="U"
4150 IF CHAR=215 THEN A$="W"
4200 RETURN
5000 REM DRAW DISPLAY
5010 CALL -936: PRINT "       YOUR LOCATION              COMPASS"
5020 PRINT : PRINT "  (BOT)   (MID)   (TOP)       REFERENCE"
5030 PRINT : TAB 34: PRINT "N"
5040 PRINT : TAB 34: PRINT "!"
5050 TAB 34: PRINT "!"
5060 TAB 29: PRINT "W <--*--> E"
5070 TAB 34: PRINT "!"
5080 TAB 34: PRINT "!"
5090 PRINT : TAB 34: PRINT "S"
5100 VTAB 6
5110 FOR K=1 TO 3
```

```
5120 PRINT : PRINT "  - - -   - - -   - - -"
5130 NEXT K
5140 VTAB 16: TAB 21: PRINT "OBSTRUCTION SENSORS"
5200 POKE 50,63
5210 VTAB 5: PRINT "                              "
5220 FOR K=1 TO 7
5230 PRINT " ";: TAB 9: PRINT " ";: TAB 17: PRINT " ";: TAB 25:PRINT "

5240 NEXT K
5250 PRINT "                          "
5300 VTAB 18: TAB 21: PRINT "                   "
5310 FOR K=1 TO 5
5320 TAB 21: PRINT " ";: TAB 39: PRINT " "
5330 NEXT K
5340 TAB 21: PRINT "                   ";
5400 VTAB 15: PRINT
5410 PRINT "                  "
5420 FOR K=1 TO 7
5430 PRINT " ";: TAB 18: PRINT " "
5440 NEXT K
5450 PRINT "                  ";
5500 POKE 50,255
5900 RETURN
9000 CALL -936: VTAB 10
9010 TAB 10: PRINT "*** GALACTI-CUBE ***"
9020 PRINT : TAB 19: PRINT "BY"
9030 PRINT : TAB 14: PRINT "ROBERT BISHOP"
9040 FOR K=1 TO 1500: NEXT K
9050 CALL -936
9110 PRINT "     YOU ARE THE CAPTAIN OF A STAR-SHIP"
9120 PRINT "EXPLORING THE OUTER LIMITS OF OUR UNI-"
9130 PRINT "VERSE.  YOU HAVE DISCOVERED A GIGANTIC"
9140 PRINT "CUBE FLOATING IN SPACE.  THROUGH THE"
9150 PRINT "ONLY OPENING YOU HAVE FLOWN YOUR SHIP"
9160 PRINT "INSIDE, BUT NOW YOU CAN'T FIND YOUR WAY"
9170 PRINT "BACK OUT!"
9190 PRINT "     FROM YOUR EXPLORATIONS YOU HAVE"
9200 PRINT "LEARNED THAT THE CUBE IS DIVIDED INTO"
9210 PRINT "AN ARRAY OF 3X3X3 CUBICAL COMPARTMENTS"
9220 PRINT "AND YOU ARE CURRENTLY IN THE CENTER-"
9230 PRINT "MOST ONE."
9250 PRINT "     YOUR SHIP IS EQUIPPED WITH A DIS-"
9260 PRINT "PLAY INDICATING YOUR LOCATION.  THE"
9270 PRINT "OBSTRUCTION SENSORS INDICATE WHICH DI-"
9280 PRINT "RECTIONS (FLASHING) ARE BLOCKED.  YOU"
9310 PRINT "MOVE YOUR SHIP BY HITTING THE FIRST"
9320 PRINT "LETTER OF THE DIRECTION YOU WANT TO GO."
9330 PRINT "YOUR FUEL SUPPLY (WHICH IS DISPLAYED BY"
9340 PRINT "HITTING THE LETTER, F) WILL ONLY LET"
9350 PRINT "YOU MAKE UP TO 40 MOVES.  GOOD LUCK!"
9999 RETURN
```

5 硬 體

簡 介

APPLE II 的彩色槍 (COLOR GUN)

——— Neil D. Lipson

APPLE II 的卡帶作業系統

——— Robert A. Stein, Jr.

利用 MICROMODEM II 傳送 BASIC 和機器語言程式

——— George J. Dombrowski, Jr.

APPLE II 的數位溫度計

——— Carl Kershner

在 APPLE II 上使用 KIM 和 SYM 格式的卡帶

——— Steven M. Welch

第五部份簡介

在一個下雨天的週末，假如沒有什麼事好做的話，最好的消遣就是爲你的APPLE裝一些硬體介面。APPLE電腦配有一些易於使用的輸入與輸出口。本章將討論如何使用它們，並提供一些有趣的計劃。

Neil Lipson的“APPLE II的彩色槍”，描述如何爲APPLE裝一個光電管介面。當它和所附的程式一起使用時，APPLE便能辨別彩色。Robert Stein的“APPLE II 的磁帶作業系統”，提供一個方法將程式歸檔命名，並存入磁帶。George Dombrowski 的“利用Micromodem II 傳送BASIC和機器語言程式”，討論如何使用一個最受歡迎的通信介面傳送程式。

Carl Kershner的“APPLE II的數位溫度計”，討論如何建立一個熱阻器（thermistor）介面，使APPLE能顯示溫度度數。最後，Steven Welch的“在APPLE II上使用KIM和SYM格式的卡帶”，使APPLE有傾印（dump）KIM-1 格式磁帶的能力，利用一個特殊的常規輸出到卡帶輸出入口。

APPLE II 的彩色槍(COLOR GUN)

作者：Neil D. Lipson

APPLE能產生許多顏色——但是如何辨別它們呢？利用一些便宜的硬體，便可以使你的APPLE變成一個彩色偵測器——可以用來自動決定任何物體的色彩。所以誰會說APPLE是色盲？

當你用“彩色槍”指向某一物體時，便可知道物體的色彩。此觀念就和電視傳送器的動作一樣。色彩分成三種顏色，即紅、藍、與黃。因此有三種輸入值到APPLE，經由paddle 0，paddle 1，與paddle 3輸入，實際上可用一個裝置算出三種顏色所佔的比率。進一步分析這比率，便可看到色彩的明暗度，利用高品質的濾光器便可精確的得到物體的色彩度數。利用此裝置，你可以將它指向一張白紙，以便調整光源的差別。因此在其範圍內的任何人工光源下，彩色槍仍可正常的執行。（例如在紅光下是不能使用的）

建立彩色槍

開始先買三個高靈敏度的硫化物光電管（直徑¼到½英吋之間）。假如三個光電管的靈敏度不一樣，則可利用軟體來改進（在程式中有說明）。只要在劃刻度程序中將彩色槍指向一張白紙（或指向一個不太亮的光源）。

彩色槍的構造很簡單。將三個光電管放在木板或其它物體上，排成三角形，每邊大約 2 英吋。再將濾光器放在光電管上，紅色在 paddle 0，藍色在 paddle 1，黃色在 paddle 2。濾光器愈好，則效果愈好。照相用的濾光器效果是最好的。然而一般的紅，藍，黃色透明塑膠便可以了。注意程式中的 REM 指述。這是為了減慢輸入的速度，避免重複輸入的現象發生。線路如圖 1 所示。

將整個架構放在大約 4 英吋長的圓筒上，圓筒內部要漆成白色。將所有東西黏好，並把漏光的地方封閉好。當線路都弄好後，便可將其插入遊戲用的 paddle。paddle 的接腳請參考使用手冊。

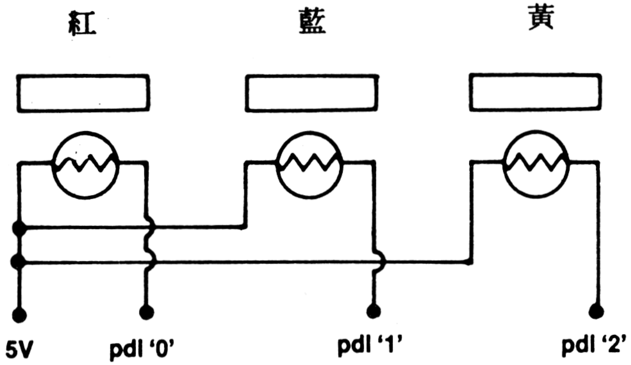


圖 1

彩色槍程式

進入APPLESOFT程式，並執行它。此彩色槍將只辨別6種顏色，當它不能確定是那一種顏色時，它會給你二種顏色（一個主要的，一個次要的）。假如色彩很純的話，便不會發生這種情況，但大部份的色彩不很純，所以要考慮這種情況。

注意在指述70 的校正方法，它用來校正藍光電管。這是因爲我所用的光電管對藍色較敏感。這由彩色槍一直說是"橘色"（藍色的補色纔便可知道。此校正方法佔了程式的大部份。假如彩色槍效果仍不好，則繼續做，直到效果好爲止。通常都不只執行一次，才會使效果很好。

適當的調整軟體，且使用最恰當的比率，你便可以決定許多其它的顏色。給予足夠的比率來選擇，你便可以給予某

物體的色彩度數（利用高品質的光電管和濾光器）。通常你可以用的照相用濾光器爲黃色（K2），紅色（25或25A），與藍色（ 47 ）。當然可能跟你買的光電管不同而有所改變。你也可以在濾光器中使用不同的顏色，只要在軟體方面適當的調整即可。儘可能買最小的濾光器（直徑約½英吋），但要保證從光電管不會漏光。假如你遵照指示做，則彩色槍便會正確的工作。祝愉快！

```
1   REM  ************************
2   REM  *                      *
3   REM  *      COLOR GUN       *
4   REM  *     NEIL LIPSON      *
5   REM  *                      *
6   REM  *  COPYRIGHT (C) 1981  *
7   REM  *   MICRO INK, INC.    *
8   REM  * CHELMSFORD, MA 01824 *
9   REM  *  ALL RIGHTS RESERVED *
10  REM  *                      *
11  REM  ************************
14  REM
15  CALL  - 936: VTAB 10: HTAB 10: PRINT "COLOR GUN BY NEIL D. LIPSON"
    FOR I = 1 TO 2000: NEXT I
17  REM  YELLOW-2
18  REM  BLUE  -1
19  REM  RED   -0
20  REM
22  CALL  - 936: PRINT : PRINT : PRINT : PRINT
25  GOSUB 1000
30  CALL  - 936: PRINT : PRINT
32 A =  PDL (0)
35  REM
40 B =  PDL (1)
45  REM
50 C =  PDL (2)
55  REM
60 A = A * A1
61 B = B * B1
62 C = C * C1
70 B = B / 1.5
100  PRINT "RED CELL=";A
110  PRINT "BLUE CELL=";B
115  PRINT "YELLOW CELL=";C
116  PRINT : PRINT
117  PRINT "THE COLOR IS:": PRINT
118  PRINT "*********************"
121  IF C < B AND C < (A) THEN  PRINT "YELLOW"
123  IF A < B AND A < C THEN  PRINT "RED"
124  IF A > B AND A > C THEN  PRINT "GREEN"
```

```
125  IF B > A AND B > C THEN  PRINT "ORANGE"
126  IF C < AC > B THEN  PRINT "PURPLE"
129  IF B < C AND B < (A) THEN  PRINT "BLUE"
130  PRINT "**********************"
131  FOR X = 1 TO 2300: NEXT X
140  GOTO 30
200  END
1000  CALL  - 936: PRINT
1010  PRINT "POINT GUN AT A WHITE SHEET OF PAPER"
1020  FOR I = 1 TO 1500: NEXT I
1030 A1 =  PDL (0)
1035  REM
1040 B1 =  PDL (1)
1045  REM
1050 C1 =  PDL (2)
1055  PRINT "A1=";A1
1056  PRINT "B1=";B1
1057  PRINT "C1=";C1
1060 D1 = A1 * B1 * C1
1070 A1 = D1 / A1
1080 B1 = D1 / B1
1090 C1 = D1 / C1
1100  PRINT "CORRECTION FACTOR FOR RED = ";A1
1110  PRINT "CORRECTION FACTOR FOR BLUE = ";B1
1120  PRINT "CORRECTION FACTOR FOR YELLOW = ";C1
1125  FOR I = 1 TO 2000: NEXT I
1130  RETURN
10000  END
```

APPLE II 的卡帶作業系統

作者: Robert A. Stein, Jr.

你是否曾經想要APPLE II 電腦系統，能利用一個檔案名稱將你的程式由庫存卡帶中載入？利用此迷你卡帶作業系統，你可以將許多程式存在一個卡帶中，並且打入名稱便可載入想要的檔案。使得你的系統不必再使用那麼多的卡帶。

此卡帶作業系統[CASSOS]存在於02C0到03FF位址上。加上選擇性的卡帶控制線路，或者購買商業成品(Candex pacific, 693 Veterans BLVD, Redwood City, CA94063)，你就不必再羨慕PET的載入技巧了。

處理程序

首先將CASSOS載入記憶體。利用CASSOS 載入一個程式，按CTRL-Y與RETURN，便會顯示“ PROG ?”

，再輸入 1 － 10 個字元的程式名稱。程式將會搜尋卡帶，假如找到程式的話，就將其載入。程式會印出“XXXXX XXXXX\LOADED”，其中 XXXXXXXXXX 表示在記憶體中的程式。假如存在有卡帶控制線路（在以後介紹），則卡帶會自動停止。假如找不到程式，則會印出一行問號（？？？？？？？？？？）。若要將一個程式寫入庫存卡帶，則打入 CTRL-Y ，“WRITE”，與RETURN。“PROG ”會要求所要寫入的程式名稱。當寫入的動作完成後，將會印出“XXXXX XXXXX OUT ”，並且會使錄音機停止。若要結束一個卡帶程式檔案，則可打入：CTRL-Y，“EOF”，RETURN ；將會寫入一個特殊起首記錄。注意，爲了保存有限的記憶體空間，EOF 常規使用程式Write 副常規，所以“XXXX XXXXXX OUT ”消息應該省略掉。

此程式輸入緩衝區的最後 63 個位置是用來存顯示的消息，所以若一次輸入超過 191 個字元，程式仍會動作，但不會有任何消息顯示。程式要在 48K 的 DOS 上執行；改變位址 0358 的值，便可允許不同的結構：

不使用	DOS	使用 DOS
1F － 8K	5F － 24K	35 － 24K
2F － 12K	7F － 32K	55 － 32K
3F － 16K	8F － 36K	65 － 36K
4F － 20K	BF － 48K	95 － 48K

程式設計

CASSOS 所用的方法是寫一個程式起首方塊（ program header block ），其中包括起首ID ，程式名稱，以及開始BASIC的載入動作。接着是程式資料，在此利用到一些APPLE監督系統常規。

一個卡帶開/關線路

下列的方塊圖描述一個簡單的線路，它能由程式控制經由APPLE II 接到錄音機的“遙控”栓停止或開啓一個錄音機。所用的理論是處理APPLE遊戲連接器的 AN3 信號。將任一值存入位址C05F將會開啓錄音機，存入C05E將會停止錄音機；它將驅動一個電晶體，打開繼電器，關閉遙控栓的連線，開啓錄音機。假如你的錄音機需要在開啓連線時才會開啓錄音機，則繼電器要關閉而不是開啓。假如你的錄音機沒有遙控能力，則可利繼電器中斷（ interrupt ）錄音機的電源，達到控制的目的。

零件表

6VDC繼電器（275－004）

NPN電晶體（2N3568）

1000歐姆電阻

250歐姆電阻

迷你一栓（Mini-plug）

所有的連線將接到DIP Header，它接着一個16脚的IC，所以在使用卡帶開啓/關閉線路時不必修改遊戲paddle。將2500歐姆的電阻和繼電器的線圈平行接起來，以便由遊戲連接器的信號驅動。假如你的錄音機不能遙控回轉（rewind），則可迴避電晶體在地線和繼電器之間加一開關，按下開關時，便可使錄音機回轉。假如只能做停止動作，則只好手動開啓錄音機。

```
 1 REM *************************
 2 REM *                       *
 3 REM *     CASSETTE C.S.     *
 4 REM *   BY ROBERT STEIN     *
 5 REM *                       *
 6 REM *       DIRECTORY       *
 7 REM *                       *
 8 REM *  COPYRIGHT (C) 1981   *
 9 REM *    MICRO INK, INC.    *
10 REM * CHELMSFORD, MA 01824  *
11 REM *  ALL RIGHTS RESERVED  *
12 REM *                       *
13 REM *************************
```

```
 14 REM
 15 REM
 16 REM
 20 N=1: CALL -936: VTAB (10): DIM X$(1)
 25 INPUT "INSERT LIBRARY TAPE AND DEPRESS 'RETURN'",X$
 30 POKE -16289,0: CALL -936: GOSUB 300
 40 PRINT "FILE #  PROGRAM NAME  BYTES"
 50 PRINT "------  ------------  -----"
 60 CALL 840: CALL -259
 70 IF PEEK (688)= ASC("E") THEN 210
 80 IF PEEK (688)= ASC("S") THEN 200
100 REM  LOAD PROGRAM INTO MEMORY BELOW THE DIRECTORY PROGRAM.
105 D= PEEK (856)-3
110 POKE 60, PEEK (700): POKE 61,( PEEK (701)-3)
120 POKE 62,255: POKE 63,D: CALL -259
130 PRINT N,: POKE 789,2: POKE 788,177: CALL 785
140 M=( PEEK (700)/2)+ PEEK (701)*128
150 L=2*(( PEEK (856)*128+128)-M):N=N+1
160 PRINT "     ";L: GOTO 60
200 GOSUB 300: PRINT "NO EOF MARK"
210 POKE -16290,0: GOSUB 300
230 PRINT : PRINT "***END OF FILE***"
240 CALL -155
300 FOR I=1 TO 30
305 L= PEEK (-16336)+ PEEK (-16336): NEXT I
310 CALL -1059: RETURN
```

```
0800      ;*************************
0800      ;*                       *
0800      ;*     CASSETTE  C.S.    *
0800      ;*    BY ROBERT STEIN    *
0800      ;*                       *
0800      ;*        CASSOS         *
0800      ;*                       *
0800      ;*  COPYRIGHT (C) 1981   *
0800      ;*    MICRO INK, INC.    *
0800      ;* CHELMSFORD, MA 01824  *
0800      ;*  ALL RIGHTS RESERVED  *
0800      ;*                       *
0800      ;*************************
0800      ;
0800      ;
0800      SLO     EPZ $3C          ;TAPE BUFFER START/END
0800      SHI     EPZ $3D
0800      ELO     EPZ $3E
0800      EHI     EPZ $3F
0800      OFFSET  EPZ $50          ;OFFSET STORAGE
0800      SAVEY   EPZ $60          ;SAVE Y-REG
0800      IN      EPZ $60          ;INPUT PARAMETERS
0800      INLO    EPZ $60
0800      INHI    EPZ $61
0800      PPL     EPZ $CA          ;INTEGER BASIC PROGRAM
0800      PPH     EPZ $CB          ; POINTER
0800      ;
0800      FCHAR   EQU $0201        ;1ST CHARACTER IN BUFFER
0800      WBUF    EQU $0200        ;WORK BUFFER
0800      IN1     EQU $02A3        ;PROG. NAME INPUT BUFFER
```

```
0800            ID      EQU $02B0          ;HEADER ID, 'S' CR
0800            NAME    EQU $02B1          ;PROGRAM NAME
0800            PEND    EQU $02BB          ;END SENTINAL (FF)
0800            PHL     EQU $02BC          ;BASIC TOP
0800            PHH     EQU $02BD
0800            ;
0800            CLRAN3  EQU $C05F          ;CLEARS GAME I/O AN3
0800            SETAN3  EQU $C05E          ;SETS GAME I/O AN3
0800            BASIC2  EQU $E003          ;INTEGER BASIC WARM START
0800            BELL    EQU $FBDD          ;MONITOR BEEP ROUTINE
0800            CR      EQU $FC62          ;MONITOR CARRIAGE RETURN
0800            GETLN2  EQU $FD6C          ;MONITOR INPUT ROUTINE
0800            COUT    EQU $FDED          ;MONITOR OUTPUT ROUTINE
0800            WRITE   EQU $FECD          ;MONITOR TAPE WRITE
0800            READ    EQU $FEFD          ;MONITOR TAPE READ
0800            ;
02C0                    ORG $2C0
02C0                    OBJ $800
02C0            ;
02C0 A9D3       PWRITE  LDA #$D3           ;SET LABEL ID TO 'S'
02C2 8DB002             STA ID
02C5 A9B1               LDA #NAME          ;OFFSET TO BUFFER
02C7 206703             JSR INIT
02CA A9FF       WEOF    LDA #$FF           ;LABEL SENTINAL
02CC 8DBB02             STA PEND
02CF A5CA               LDA PPL            ;STORE TOP OF PROGRAMADDRESS
02D1 8DBC02             STA PHL
02D4 A5CB               LDA PPH
02D6 8DBD02             STA PHH
02D9 20CDFE             JSR WRITE          ;WRITE LABEL
02DC A4CA               LDY PPL
02DE A5CB               LDA PPH
02E0 206003             JSR SETS           ;SET TOP WRITE/HIMEM BOTTOM
02E3 20CDFE             JSR WRITE          ;WRITE PROGRAM
02E6 A9EB               LDA #OUT           ;SET TO WRITTEN MESSAGE
02E8 207E03             JSR ECHO           ;PRINT XXXXXXXXXX OUT
02EB            ;
02EB 87A0CF     OUT     HEX 87A0CFD5D4FF   ;" OUT" MESSAGE
02EE D5D4FF
02F1 87A0CC     LOADED  HEX 87A0CCCFC1C4C5C4FF ;" LOADED" MESSAGE
02F4 CFC1C4
02F7 C5C4FF
02FA D0D2CF     PROG?   HEX D0D2CFC7BFFF   ;" PROG?" MESSAGE
02FD C7BFFF
0300
0300 A202       TYPE3   LDX #$02           ;SET HI ADDRESS TO 02
0302 D007               BNE TYPE           ;BRANCH TO MAIN ROUTINE
0304 8460       NLTYPE  STY SAVEY
0306 2062FC             JSR CR             ;OUTPUT CR/LF
0309 A460               LDY SAVEY          ;RESTORE Y
030B 8E1503     TYPE    STX CONT+2         ;MODIFY LOAD INSTRUCTION
030E 8C1403             STY CONT+1
0311 A000               LDY #$00           ;SET I-VALUE
0313 B9FA02     CONT    LDA PROG?,Y        ;GET CHARACTER
0316 C9FF               CMP #$FF           ;DELIMETER?
0318 F02D               BEQ TDONE          ;YES- RETURN
031A 20EDFD             JSR COUT           ;OUTPUT
031D C8                 INY                ;INCREMENT INDEX
```

```
031E D0F3               BNE CONT        ;CONTINUE (JMP)
0320
0320 48         INPUT   PHA             ;SAVE INPUT COUNT
0321 A902               LDA /IN1        ;SET HI INPUT ADDRESS
0323 8660               STX INLO        ;STORE ADDRESS
0325 8561               STA INHI        ;(PHA & LDA TO CHG HI)
0327 A9A0               LDA #$A0        ;SET PROMPT TO " "
0329 206CFD             JSR GETLN2      ;INPUT TO COMMON BUFFER
032C 68                 PLA             ;RESTORE COUNT
032D AA                 TAX             ;SET TO X
032E A000               LDY #$00        ;SET Y-INDEX
0330 B90002     MOVE    LDA WBUF,Y      ;LOAD FROM WORK BUFFER
0333 C98D               CMP #$8D        ;LAST INPUT?
0335 F008               BEQ CR1         ;YES
0337 9160               STA (IN),Y      ;STORE IN USER AREA
0339 C8                 INY             ;INCREMENT POINTER
033A CA                 DEX             ;DECREMENT COUNTER
033B F00A               BEQ TDONE       ;RETURN IF DONE
033D D0F1               BNE MOVE        ;ELSE BRANCH TO LOOP
033F A9A0       CR1     LDA #$A0
0341 9160               STA (IN),Y      ;SPACE FILL
0343 C8                 INY
0344 CA                 DEX
0345 D0F8               BNE CR1         ;LOOP TILL MAXIMUM
0347 60         TDONE   RTS             ;RETURN
0348            ;
0348 A0B0       SLBL    LDY #ID         ;SET ID LABEL ADDRESS
034A A200               LDX #$00        ;SET START FLAG
034C 205103             JSR SEC         ;SET-UP TO SET END TOO
034F A0BD               LDY #PHH        ;SET END OF LABEL
0351 A902       SEC     LDA #$02
0353 D004               BNE SET         ;BRANCH TO SET START
0355 A0FF       SHIM    LDY #$FF        ;SET HIMEM:
0357 A995               LDA #$95        ;(CHANGE FOR MORE MEMCRY)
0359 953D       SET     STA SHI,X       ;SET START
035B 943C               STY SLO,X       ; OR END
035D E8                 INX             ;BUMP END BY 2 FOR
035E E8                 INX             ;END PAIR
035F 60                 RTS
0360 A200       SETS    LDX #$00
0362 205903             JSR SET         ;SET BASIC TOP & BOTTOM
0365 D0EE               BNE SHIM
0367            ;
0367 8550       INIT    STA OFFSET      ;STORE INBUF OFFSET
0369 A202               LDX /PROG?      ;SET " PROG?" ADDRESS
036B A0FA               LDY #PROG?
036D 20C403             JSR NLTYPE      ;OUPUT WITH NL
0370 204803             JSR SLBL        ;SET LABEL PARAMETERS
0373 A90A               LDA #$0A        ;INPUT = 10 CHARACTERS
0375 A650               LDX OFFSET      ;USER INPUT OFFSET
0377 202003             JSR INPUT       ;INPUT PROGRAM NAME
037A 8D5FC0             STA CLRAN3      ;TURN ON CASSETTE
037D 60                 RTS
037E            ;
037E 48         ECHO    PHA             ;STORE OFFSET
037F 8D5EC0             STA SETAN3      ;TURN OFF CASSETTE
0382 A202               LDX /NAME       ;SET TO OUTPUT LABEL NAME
```

```
0384 ACB1           LDY #NAME
0386 200403         JSR NLTYPE
0389 68             PLA                 ;GET MESSAGE
038A A8             TAY                 ;PUT IN Y FOR TYPE
038B 200003         JSR TYPE3           ;OUTPUT " OUT" OR " LOADED"
038E 4C03E0         JMP BASIC2
0391           ;
0391 A9A3      PLOAD  LDA #IN1          ;INPUT PROGRAM NAME
0393 206703         JSR INIT            ; TO IN1 ($2A3)
0396 204803    TRYAGN JSR SLBL          ;SET LABEL PARAMS.
0399 20FDFE         JSR READ            ;READ LABEL
039C ADB002         LDA ID              ;GET ID
039F C9D3           CMP #"S"
03A1 D029           BNE NFOUND          ;EOF OR NOT ON TAPE
03A3 ACBC02         LDY PHL
03A6 ADBD02         LDA PHH
03A9 206003         JSR SETS            ;READ PROGRAM PARAMETERS
03AC 20FDFE         JSR READ            ;READ PROGRAM
03AF A200           LDX #$00            ;SET INDEX
03B1 BDB102    TEST   LDA NAME,X        ;COMPARE FOUND NAME
03B4 DDA302         CMP IN1,X     ; WITH INPUT NAME
03B7 D0DD           BNE TRYAGN
03B9 E8             INX
03BA E00A           CPX #$0A      ;CHECK ALL LOOKED AT
03BC D0F3           BNE TEST
03BE ADBC02         LDA PHL       ;SET TOP OF BASIC ADDRESS
03C1 85CA           STA PPL
03C3 ADBD02         LDA PHH
03C6 85CB           STA PPH
03C8 A9F1           LDA #LOADED   ;SET TO " LOADED"
03CA D0B2           BNE ECHO      ;OUTPUT WITH VERIFY NAME
03CC 8D5EC0    NFOUND STA SETAN3  ;TURN OFF CASSETTE
03CF A220           LDX #$20
03D1 A9BF      NC     LDA #$BF    ;PRINT ??????????
03D3 20EDFD         JSR COUT
03D6 CA             DEX
03D7 D0F6           BNE NC        ;LOOP
03D9 20DDFB         JSR BELL      ;SOUND TONE
03DC F0B3           BEQ PLOAD     ;RETURN FOR NEW NAME
03DE           ;
03DE AD0102    WHICH  LDA FCHAR   ;FIRST CHAR OF FUNCTON (E,R,W)
03E1 C9D7           CMP #"W"      ;"WRITE"
03E3 F010           BEQ SAVE
03E5 C9C5           CMP #"E"      ;"EOF"
03E7 D0A8           BNE PLOAD     ;"READ"
03E9 8DB002         STA ID        ;STORE E AS ID IN LABEL
03EC 204803         JSR SLBL      ;SET LABEL PARAMETERS
03EF 8D5FC0         STA CLRAN3    ;TURN ON CASSETTE
03F2 4CCA02         JMP WEOF      ;BRANCH TO WRITE EOF
03F5 4CC002    SAVE   JMP PWRITE  ;BRANCH TO WRITE PROGRAM
03F8           ;
03F8 4CDE03    CTRLY  JMP WHICH   ;CONTROL-Y TRANSFER TO CHECK FN
03FB 0000      NMI    HEX 0000    ;NMI VECTOR
03FD 0000      IRQ    HEX 0000    ;IRQ VECTOR
03FF 00             HEX 00
                    END

***** END OF ASSEMBLY
```

```
**************************
*                        *
* SYMBOL TABLE -- V 1.5 *
*                        *
**************************

LABEL. LOC.   LABEL. LOC.   LABEL. LOC.

** ZERO PAGE VARIABLES:

SLO     003C  SHI     003D  ELO     003E  EHI     003F  OFFSET  0050  SAVEY  0060
IN      0060  INLO    0060  INHI    0061  PPL     00CA  PPH     00CB

** ABSOLUTE VARABLES/LABELS

FCHAR   0201
WBUF    0200  IN1     02A3  ID      02B0  NAME    02B1  PEND    02BB  PHL    02BC
PHH     02BD  CLRAN3  C05F  SETAN3  C05E  BASIC2  E003  BELL    FBDD  CR     FC62
GETLN2  FD6C  COUT    FDED  WRITE   FECD  READ    FEFD  PWRITE  02C0  WEOF   02CA
OUT     02EB  LOADED  02F1  PROG?   02FA  TYPE3   0300  NLTYPE  0304  TYPE   030B
CONT    0313  INPUT   0320  MOVE    0330  CR1     033F  TDONE   0347  SLBL   0348
SEC     0351  SHIM    0355  SET     0359  SETS    0360  INIT    0367  ECHO   037E
PLOAD   0391  TRYAGN  0396  TEST    03B1  NFOUND  03CC  NC      03D1  WHICH  03DE
SAVE    03F5  CTRLY   03F8  NMI     03FB  IRQ     03FD

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:01E2
```

利用Micromodem II 傳送BASIC和機器語言程式

作者：George J. Dombrowski, Jr.

D. C. Hayes Micromodem是APPLE 最受歡迎的通信介面。利用此介面，你可以在你的APPLE 和遠方的電腦傳送程式。下面將提供一些常規，使二部APPLE 之間能傳送BASIC和機器語言程式。

不可否認的，D. C. Hayes 的Micromodem是一個非常複雜的遠距離通信裝置，幾個月前我購買了一個Micromodem，我覺得它的性能非常好。將此裝置直接和Ma Bell 連結時，你可以很容易的設計，讓它自動的替你接電話，或者傳送一些短的消息到其它機器。

D. C. Hayes Associates 最好的特色之一是在 85 頁的使用手册上都有詳細說明並附有程式範例。然而不管使用手册有多好，它有一個明顯的遺漏。我原來購買Micromodem II ，是因爲它能很容易的在二部APPLE 電腦之間傳送BASIC 和機器語言程式。雖然使用手册上說明APPLE 電腦的Datamover 程式能適用於Micromodem的韌體（firmware），但將BASIC 程式送到另一部電腦的較簡單且直

接的方法並沒有提到。此篇文章將描述一個直接的程序來傳送BASIC程式，並且提供一個ARRLESOFT 常規將機器語言程式或二元化資料送到另一部APPLE II。

利用Micromodem II 將一個BASIC程式以直接的方式送出是一件簡單的事。一旦把線接好後，打入CTRL R，接着打RR #S，其中S是調變/解調器（modem）擴充接點（slot）號碼，便可把接收資料的電腦變成遙控（remote）形式。當出現BASIC系統符號時，表示已達到遙控的狀況。此時接收資料的電腦便等待輸入資料。它可以接收由自己的鍵盤所打入的資料，你的鍵盤打入的資料，或者是在執行程式過程中由你的電腦所產生的資料。換句話說，接收資料的電腦將會接收由另一部電腦所送出的程式，並且把每一行當作一個命令。在送出程式之前，首先要在二部電腦之間執行一些步驟，以便做傳送動作。

一旦把接收資料的電腦設爲遙控形式後，必須再利用INT或FP DOS命令進入BASIC。此時遙控的電腦必須執行PR #0 以便將輸出送到視頻輸出入口。這是一個預防步驟，它能防止接收資料的電腦命令翻譯器產生一些額外的訊號。因爲這些訊號可能會被送出資料的電腦接收，而干擾程式的傳送。在執行此命令後，送出資料的電腦操作者才會看到BASIC系統符號。爲了在你的電腦上列印（List）程式，必須打入CTRL-A/CTRL-X 停止終端機模式（terminal mode）。接收資料的電腦此時在遙控模式等待輸入資料，而送出資料的電腦準備列印程式。

雖然這些程序看起來很複雜，但用過幾次後會很容易記得。對於那些想坐着看機器是如何工作的人，下列的程式會產生一個EXEC檔案達到此目的。

從現在開始，在本地（ local ）鍵盤打入的命令將不會送到遠方的機器。首先必須利用POKE 1912 + S, 18，緊接着POKE 1528 + S, 80, 設立送出資料的Return遲延軔體（ firmware ）。在Return之後的暫停時間能允許接收資料的機器在下一行資料送出之前翻譯並執行每一行資料。暫存器1528 + S 平常在終端機模式時是3，表示遲延 30 微秒。第二，載入要送出的程式，並打入POKE 33, 30 使LIST編排常規不能工作。最後，打入PR #2 ，並且在游標（ cursor ）出現後，再打LIST命令。

APPLE此時在遙控模式等待輸入，而送出資料的電腦準備LIST程式。

執行此程式會產生一個EXEC檔案，然後再載入所要送出的程式。最後，EXEC BASIC PROGRAM TRANSFER。此EXEC檔案將和BASIC一起工作。當傳送完畢後使用者的機器會在終端機模式。接着必須送PR #2 到遙控的電腦，以便接收它的輸出。

二元化資料或機器語言程式也可用類似的方法來傳送，只要把監督系統的十六進位傾印（ dump ）常規修改即可。平常打入RETURN 之後，此常規會顯示一個十六進位的位址並緊接着一個“ - ”符號。這一定要修改的，因為在輸入二元化資料時，監督系統翻譯器要求在位址之後要緊接着一

個“：”符號。我們可以把一小部份的F8 ROM晶片（$FD92－$FDC5）重新載入到RAM $1000－$1033的位址上。位址$100D的$A0（“－”）要改成$BA（“：”）。除此之外，在$1021－$1023 JSR 指令的位址要由$FD92改成$1000。此HEX 傾印常規已經合併到一個能處理上述瑣碎雜務的APPLESOFT BASIC程式。

利用Micromodem Ⅱ傳送APPLESOFT的二元化資料

雖然這些方法需要一些軟體，也很容易做，但它們有一個缺點。即傳送BASIC和機器語言程式所需的時間大於（大約分別為20％與130％）由程式長度所算的時間。這是因為整數BASIC和APPLESOFT 程式是以保留字符號存在記憶體中。例如PRINT, POKE, 或NEXT 在記憶體中只需要一個位元組。在傳送速率（baud）為300時送一個位元組要1/30秒；然而利用上述的LIST 程序，傳送一個保留字，如PRINT大約需要5/30秒。

同樣的，在傳送機器語言程式方面，每傳送8個位元組的資料有4位十六進位位址，冒號，8對十六進位資料，以及8個空白須要傳送。所以在記憶體中每8個位元組須要傳送29個字元。

不管上述的缺點，在短矩離，程式不很大的情況下，這個技巧非常的方便，且所費的時間不多。

注意：使用此程式時，Micromodem 要選第 2 個擴充接點。假如選用其它擴充接點，則暫存器 1530 和 1914 要改成 1528 + S 與 1912 + S，其中 S 是擴充接點號碼。

```
1   REM  ************************
2   REM  *                      *
3   REM  * MICROMODEM TRANSFERS *
4   REM  *   GEORGE DOMBROWSKI  *
5   REM  *                      *
6   REM  *    BINARY TRANSFER   *
7   REM  *                      *
8   REM  *  COPYRIGHT (C) 1981  *
9   REM  *    MICRO INK, INC.   *
10  REM * CHELMSFORD, MA 01824 *
11  REM *  ALL RIGHTS RESERVED *
12  REM *                      *
13  REM ************************
14  REM
15  REM
19  REM  BINARY TRANSFER/MICROMODEM II
20 D$ =  CHR$ (4)
30  PRINT D$"NOMON C,I,O"
40  GOSUB 420
50  INPUT "IS RECEIVING COMPUTER IN REMOTE MODE WITH EITHER BASIC INITIAL
     IZED?";ANS$
60  PRINT
70  IF  LEFT$ (ANS$,1) <  > "Y" THEN  PRINT "TRANSFER ADANDONED": END
80  POKE 1530,60: POKE 1914,18: REM  600 MSEC WAIT AFTER CARRIAGE RETURN.
     AUTO LINE FEED IS ACIVATED AND THE WAIT FUNCTION + LOCAL DISPLAY ENA
     BLED.
90  PRINT "STARTING ADDRESS-": INPUT "(MUST END WITH 0 OR 8)";ST$
100  REM  LINES 110/170 - HEXIDECIMAL TO DECIMAL CONVERSION.
110 Z$ = "0123456789ABCDEF"
120  FOR I =  LEN (ST$) TO 1 STEP  - 1
130  FOR J = 1 TO  LEN (Z$)
140  IF  MID$ (Z$,J,1) <  >  MID$ (ST$,I,1) THEN  NEXT J
150 DEC = DEC + (J - 1) * (16 ^ X)
160 X = X + 1: NEXT I
170 HB =  INT (DEC / 256):LB = DEC - (HB * 256)
180  REM  LINE 190 PLACES THE DECIMAL EQUIVALENTS OF THE HIGH & LOW BYTE
     ADDRESS INTO THE PAGE 0 LOCATIONS USED BY THE MEMORY DUMP ROUTINE.
190  POKE 61,HB: POKE 60,LB
200  INPUT "NUMBER OF BYTES (DECIMAL) ";NB
210  PRINT : INVERSE : HTAB 6: PRINT "HITTING ANY KEY ABORTS TRANSFER":
     NORMAL
220  PRINT D$"IN #0"
230  PRINT D$"PR #2"
240  PRINT "CALL-151"
250  PRINT : REM  SENDS CARRIAGE RETURN.
260  FOR I = 1 TO  INT (NB / 8) + 1
270  IF  PEEK ( - 16384) > 127 THEN  POKE  - 16368,0: GOTO 300
280  CALL 4113: REM  CALLS MACHINE LANGUAGE ROUTINE BELOW.
290  NEXT I
```

```
300  PRINT
310  PRINT "3DOG"
320  PRINT D$"PR #0"
330  PRINT
340  POKE 1530,3: REM  NORMAL 30 MSEC WAIT
350  PRINT "            *** ALL DONE ***"
360  PRINT : PRINT "THE SENDING COMPUTERIS NOW IN TERMINAL MODE & THE REC
     EIVING COMPUTER HAS BEEN RETURNED WITH BASIC UP IN REMOTE MODE."
370  PRINT : INVERSE : HTAB 15: PRINT "HIT RETURN": NORMAL
380  PRINT D$"IN #2"
390  POKE 1914,138: REM  INITIATE TEMINAL MODE/FULL-DUPLEX (USE 10 FOR
     HALF-DUPLEX).
400  END
410  REM  LINES 420/450 LOAD RELOCATED MEMORY DUMP ROUTINE AT $1000.
420  FOR M = 4096 TO 4147: READ D: POKE M,D: NEXT M
430  RETURN
440  DATA  164,61,166,60,32,142,253,32,64,249,160,0,169,186,76,237,253,16
     5,60,9,7,133,62,165,61,133,63,165,60,41,7,208,3,32,0,16
450  DATA  169,160,32,237,253,177,60,32,218,253,32,186,252,144,232,96
460  REM   THE BASIC PRGM + DUMP ROUTINE OCCUPY $800-$1040. IF THE BINARY
     DATA TO BE SENT RESIDES IN THIS RANGE, IT MUST FIRST BE RELOCATED  W
     ITH THE MONITOR MOVE COMMAND.
```

```
1   REM  ************************
2   REM  *                      *
3   REM  * MICROMODEM TRANSFERS *
4   REM  *   GEORGE DOMBROWSKI  *
5   REM  *                      *
6   REM  *    BASIC TRANSFER    *
7   REM  *                      *
8   REM  *  COPYRIGHT (C) 1981  *
9   REM  *    MICRO INK, INC.   *
10  REM  * CHELMSFORD, MA 01824 *
11  REM  *  ALL RIGHTS RESERVED *
12  REM  *                      *
13  REM  ************************
14  REM
15  REM
16  REM  BASIC TRANSFER/MICROMODEM II
20  REM  FIRST RUN THIS PROGRAM AND THEN
30  REM  ESTABLISH REMOTE CONTROL OF RECEIVING MACHINE
40  REM  LEAVE TERMINAL MODE BY TYPING CTRL-A/CTRL-X
50  REM  THEN TYPE <EXEC BASIC PROGRAM TRANSFER>
60 D$ =  CHR$ (4)
70  PRINT D$"OPEN BASIC PROGRAM TRANSFER"
80  PRINT D$"WRITE BASIC PROGRAM TRANSFER"
90  PRINT "POKE 1530,80:REM FOR LONG FLOATING POINT PROGRAMS A GREATER DE
     LAY MAY BE REQUIRED."
100  PRINT "POKE 1914,18"
110  PRINT "POKE 33,30"
120  PRINT "IN#0"
130  PRINT "PR#2"
140  PRINT "LIST"
150  PRINT "PR#0"
160  PRINT "IN#2"
170  PRINT "TEXT"
180  PRINT "POKE 1530,3"
190  PRINT "POKE 1914,138"
200  PRINT D$"CLOSE"
210  END
```

APPLE Ⅱ 的數位溫度計

作者：Carl Kershner

APPLE II 能夠測知溫度嗎？熱阻器（ thermistor ）的探測針直接接到APPLE II的輸入／輸出連接器，將其結果經過線性運算之後，便可產生華氏或攝氏度數。這篇報告將說明如何做。

熱阻器的溫度探測針可以利用遊戲用的輸入／輸出連接器直接接到APPLE II 電腦。這是可以的，因爲熱阻器是“熱電阻”，當溫度改變時，它能顯示大的電阻變化。在APPLE的paddle輸入口，PDL（0，1，2，&3），本來就是一個對這種可變電阻的A/D轉換器。

APPLE和熱阻器非常的配合，因爲熱阻器的非線性性質可以用簡單的軟體來解決。除此外，APPLE的533定時器（timer）中的RC網路所消耗的小電流剛好符合熱阻器所要求的零一功率（zero-power）條件。使用熱阻器時，非線性性質與探測針電流的感應溫度都是不易處理的性質。

此程式在APPLESOFT上做，其分輸入節，資料換算節，與顯示節。輸入節要求選擇paddle輸入與大部份廠商公

認的二種熱阻器規格：RO，量室溫的電阻；RA ，用一個值表示在 25°C 與在 50°C 的電阻比率。接着用所選擇的 paddle輸入口讀入，並依比率表示電阻值。從溫度－電阻的關係式便可算出相對的華氏與攝氏溫度：

$$R_1 / R_2 = e^{B(1/T_1 - 1/T_2)}$$

其中 R_1 與 R_2 分別是在絕對溫度 T_1 與 T_2 的電阻值，β 是一個常數，依不同質料的熱阻器而不同。結果將四捨五入取整數值，並以三位數顯示出來，若有居先零（ leading zeros ）以空白取代，零度以下以負號表示。

要將熱阻器接到APPLE II ，首先將探測針的一脚接到＋5V，即Game I/O 連接器的第 1 脚，再將另一脚接到 Game I/O 連接器的第 6, 7, 10, 或 11脚。如果選用室溫電阻，且其比率值適合所需的溫度範圍與靈敏度，就不必再做任何修改。一個 40,000 歐姆的熱阻器，比率值為 9 或 10 ，將提供最少華氏一度的靈敏度，並且適用於室溫的範圍。若想要對你的特殊應用選擇一個適用的熱阻器，最好的方法是執行此程式，把遊戲的 paddle 當作輸入，打入RO與RA 的規格，然後觀察其適用溫度範圍與靈敏度。這步驟是強調此程式額外的好處，可用來幫助你選擇適用的熱阻器。

在此所用的熱阻器其價錢少於美金 5 元，可以從供應站或直接從工廠購得。 Fenwal GA44P2探測針型式的熱阻器，有一 40,000 歐姆的室溫電阻，比率值為 9.53 ，適用於

室溫的測量，而Fenwal GA42P2有一15,000 歐姆的電阻，比率值爲9.1，適用於室內或室外。最好將探測針置於金屬管內，以免損壞，並提供溫度慣性，使得短暫的溫度變化最小。如果想要求準確度高一點，則最好將探測針和實驗用溫度計對照刻度比較，因爲對於便宜的探測針，其RO或RA值均不會比± 10 %還好。

因爲熱阻器有相當高的電阻，傳輸線，且接觸溫度的影響也可不管，所以探測針若遠離電腦控制台仍可正常操作。所以APPLE II 數位溫度計可適用於室內或室外的溫度監督工作。

本章所提的Fenwal 產品可購於Fenwal Electronics, 63 Fountain St., PO BOX 585, Framingham, MA 01701。

```
10   REM  ************************
15   REM  *                      *
20   REM  *  DIGITAL THERMOMETER *
25   REM  *     CARL KERSHNER    *
30   REM  *                      *
35   REM  *      THERMOMETER     *
40   REM  *                      *
45   REM  *  COPYRIGHT (C) 1981  *
50   REM  *    MICRO INK, INC.   *
55   REM  * CHELMSFORD, MA 01824 *
60   REM  *  ALL RIGHTS RESERVED *
65   REM  *                      *
70   REM  ************************
80   REM
90   REM
100 REM   DIGITAL THERMOMETER FOR THERMISTOR PROBE(DISPLAYS BOTH CELCIUS
     &FAHRENHEIT)
110 PRINT "WHICH INPUT DO YOU WANT(0,1,2,3)": INPUT NUMBER
120 PRINT "WHAT THERMISTOR CONSTANTS DO YOU WANT(RO,RATIO)": INPUT RO,RA

125 BETA = 1.7636E3 *  LOG (RA)
130  HOME : REM   CLEAR SCREEN
140  REM  PRINT TEMPERATURE SCALE CHARACTERS
```

```
150  GR : COLOR= 15
160 HLIN 26,27 AT 6: HLIN 26,27 AT 7: HLIN 26,27 AT 9: HLIN 26,27 AT 10:
     VLIN 7,9 AT 25: VLIN 7,9 AT 28
170 HLIN 34,38 AT 9: HLIN 34,38 AT 10: HLIN 34,36 AT 14: HLIN 34,36 AT 1
    5: VLIN 9,20 AT 33
180 HLIN 26,27 AT 23: HLIN 26,27 AT 24: HLIN 26,27 AT 26: HLIN 26,27 AT
    27: VLIN 24,26 AT 25: VLIN 24,26 AT 28
190 VLIN 28,29 AT 38: VLIN 27,28 AT 37: VLIN 26,27 AT 36: VLIN 26,27 AT
    35: VLIN 27,28 AT 34
200 VLIN 28,35 AT 33: VLIN 35,36 AT 34: VLIN 36,37 AT 35: VLIN 36,37 AT
    36: VLIN 35,36 AT 37: VLIN 34,35 AT 38
210 T = 298: REM   SET T(0) AT 298 DEGREES ABSOLUTE
220 RI = 589.94 *  PDL (NUMBER): REM   READ INPUT & SCALE TO OHMS
230  IF RI = 0 THEN RI = 1: REM   PREVENT DIVISION BY ZERO
240 TC =  INT (1 / (1 / T -  LOG (RO / RI) / BETA) - 272.5): REM CALCUL
     ATE TEMPERATURE IN DEGREES CELCIUS AND ROUND TO NEAREST INTEGER
245  IF  ABS (TC) > 999 THEN  GOTO 220: REM   LIMIT OVERFLOWING DISPLAY
250 SIGN = 0
260  IF TC < 0 THEN SIGN = 15
270  COLOR= SIGN
280  HLIN 3,5 AT 29: HLIN 3,5 AT 30: REM   DISPLAY NEGATIVE SIGN
290 TC =  ABS (TC)
300 J =  INT (TC / 100):I = J: REM   SEPARATE HUNDRED'S DIGIT
310  IF J = 0 THEN J = 10: REM   BLANK LEADING ZERO
320 X = 1:Y = 26: GOSUB 1000: REM   DISPLAY CELCIUS HUNDRED'S
330 J =  INT ((TC - J * 100) / 10): REM   SEPARATE TEN'S DIGIT
340  IF I = 0 AND J = 0 THEN J = 10: REM BLANK BOTH HUNDRED'S AND TEN'S
      LEADING ZEROS IF J&I ARE BOTH ZERO
350 X = 9:Y = 26: GOSUB 1000: REM   DISPLAY CELCIUS TEN'S DIGIT
360 J = TC - I * 100 - J * 10: REM   SEPARATE ONE'S DIGIT
370 X = 17:Y = 26: GOSUB 1000: REM   DISPLAY CELCIUS ONE'S DIGIT
380 TF =  INT (9 * (1 / (1 / T -  LOG (RO / RI) / BETA) - 273) /5 + 32.5
     ): REM   CALCULATE FAHRENHEIT & ROUND TO NEAREST INTEGER
390 SIGN = 0
400  IF TF < 0 THEN SIGN = 15
410  COLOR= SIGN
420  HLIN 3,5 AT 12: HLIN 3,5 AT 13: REM   DISPLAY NEGATIVE SIGN
430 TF =  ABS (TF)
440 J =  INT (TF / 100):I = J: REM  SEPARATE HUNDRED'S DIGIT
450  IF J = 0 THEN J = 10: REM   BLANK LEADING ZERO
460 X = 1:Y = 9: GOSUB 1000: REM   DISPLAY FAHRENHEIT HUNDRED'S DIGIT
470 J =  INT ((TF - J * 100) / 10): REM   SEPARATE TEN'S DIGIT
480  IF I = 0 AND J = 0 THEN J = 10: REM BLANK BOTH HUNDRED'S AND TEN'S
      LEADING ZEROS
490 X = 9:Y = 9: GOSUB 1000: REM   DISPLAY FAHRENHEIT TEN'S DIGIT
500 J = TF - I * 100 - J * 10: REM   SEPARATE ONE'S DIGIT
510 X = 17:Y = 9: GOSUB 1000: REM   DISPLAY FAHRENHEIT ONE'S DIGIT
520  GOTO 220
1000  REM   SEVEN SEGMENT ENCODER
1010  ON J GOTO 1110,1120,1130,1140,1150,1160,1170,1180,1190,1200
1100 A = 15:B = 15:C = 15:D = 15:E = 15:F = 15:G = 0: GOTO 2000
1110 A = 0:B = 15:C = 15:D = 0:E = 0:F = 0:G = 0: GOTO 2000
1120 A = 15:B = 15:C = 0:D = 15:E = 15:F = 0:G = 15: GOTO 2000
1130 A = 15:B = 15:C = 15:D = 15:E = 0:F = 0:G = 15: GOTO 2000
1140 A = 0:B = 15:C = 15:D = 0:E = 0:F = 15:G = 15: GOTO 2000
1150 A = 15:B = 0:C = 15:D = 15:E = 0:F = 15:G = 15: GOTO 2000
1160 A = 15:B = 0:C = 15:D = 15:E = 15:F = 15:G = 15: GOTO 2000
1170 A = 15:B = 15:C = 15:D = 0:E = 0:F = 0:G = 0: GOTO 2000
```

```
1180 A = 15:B = 15:C = 15:D = 15:E = 15:F = 15:G = 15: GOTO 2000
1190 A = 15:B = 15:C = 15:D = 15:E = 0:F = 15:G = 15: GOTO 2000
1200 A = 0:B = 0:C = 0:D = 0:E = 0:F = 0:G = 0:J = 0: GOTO 2000
2000  REM  SEVEN SEGMENT DISPLAY
2010  COLOR= A
2020  HLIN X + 1,X + 4 AT Y
2030  HLIN X + 1,X + 4 AT Y + 1
2040  COLOR= G
2050  HLIN X + 1,X + 4 AT Y + 5
2060  HLIN X + 1,X + 4 AT Y + 6
2070  COLOR= D
2080  HLIN X + 1,X + 4 AT Y + 10
2090  HLIN X + 1,X + 4 AT Y + 11
2100  COLOR= F
2110  VLIN Y + 1,Y + 5 AT X
2120  COLOR= B
2130  VLIN Y + 1,Y + 5 AT X + 5
2140  COLOR= E
2150  VLIN Y + 6,Y + 10 AT X
2160  COLOR= C
2170  VLIN Y + 6,Y + 10 AT X + 5
2180  RETURN
```

在APPLE II 上使用KIM和SYM格式的卡帶

作者： Steven M. Welch

現在你可以經由錄音機的輸入／輸出，在APPLE和AIM，SYM，或KIM之間交換程式與資料。

大部份的KIM或SYM所有者都因爲他們的需要與價錢的便宜，都改用較好的6502 系統。假如你是其中之一，因爲APPLE的易於使用，而把你的 KIM 放着不用，利用此程式，你可以把你的APPLE II 當作“主計算機”，在此發展組合語言程式，然後再把寫好的程式“送入”你的單一面板計算機（single board computer（SBC））。利用這種方式來寫組合語言程式看不出來有何價值，直到你要用APPLE爲SYM或KIM寫一個相當大的程式才看得出來。所以用手編譯組合語言的痛苦都消失了。

在後面的程式是爲了滿足一個需要：在SYM 上寫一個大程式。我估計過若利用APPLE寫一個1500 位元組的程式，大約會節省2個人一月的時間。同時我們也有一個非常好用的組合語言編譯器可以用，而不需要辛苦的去寫機器代

碼。以前我在Colocrado大學工作時，我們利用SYM－1電腦發展一套微處理機控制的Charge Coupled Photo Diode（CCPD）攝譜儀探測器。雖然SYM－1是一個非常好的SBC，但它卻乏用來發展程式的一些功能，例如，快又大的記憶體空間，組合語言編譯器，編校程式，ASCII鍵盤，與顯示裝置。若沒有這些便利的功能，寫這個控制程式可能要花很多的時間。

為了擁有這些功能，必須購買一套微處理機發展系統（Microprocessor Development System），通常是$10－20,000。而我們並沒有這麼多錢，我們只好在APPLE II上做。APPLE幾乎擁有微處理機發展系統的大部份功能，除了它能以問答方式和SBC通信。APPLE如何和SYM交談呢？很幸運的，二部電腦都使用6502微處理機，所以在APPLE上編譯過的組合語言程式能夠在SYM或KIM上執行。而且它們的輸入與輸出都用卡帶。但不用說，它們的磁帶格式不一樣。所以我們必須做一些轉換工作，使SYM能在APPLE上做，或是APPLE能在SYM上做。因為在APPLE上比較容易寫程式，所以我決定使我的APPLE卡帶能在SYM上執行。

這兒有另一個理由使APPLE的卡帶能在SYM上執行：製造SYM的人－SYNERTEK，在使用手册的後面附有SYM的監督程式。在此監督程式中，含有用來產生SYM或KIM磁帶的常規。我們可以很容易的修改此程式，然後在APPLE上執行。不需要用到計時器（timer）（APPLE

也沒有），序列（ serial ）的資料由6522輸出口送出。雖然APPLE沒有6522 ，但它有許多單一位元輸出口，我們只要將一個單一位元輸出口調整其輸出標準（ level ），便可當做一個錄音機的介面了。雖然它不是一個6522 的輸出口，但在某些情況下可當作它是。APPLE所做的方式爲，任何時間卡帶輸出口的位址出現在位址巴士（ address bus ）時，便改變卡帶輸出正反器（ flip-flop ）的狀態。另一方面，在SYM 我們送一個特殊位元到一個位址，並且這些位元會出現在輸出閂鎖中（ output latch ）。

基本上，我們可以假設APPLE的卡帶就是 SYM的卡帶輸出。各位應該注意，APPLE並不能控制輸出信號的相位（ phase ），但大部份的卡帶輸入常規對於相位均不會太靈敏。很幸運的，多虧運氣好或是SYNERTEK的程式設計師精心策劃，90 %的卡帶輸出都是用這種方式。這個特色使得此程式能馬上爲APPLE使用。一旦我從SYNERTEK碼取出合適的部份，並想出它們做些什麼，則我只要改變幾行便可得到列印在後面的程式，我將不說明它如何工作，但是我在程式中已加了不少的說明，有興趣者可自行看看。

如何使用此程式

首先做一個SYNC磁帶是一個很好的主意。APPLE輸出標準大約是 SYM輸出標準的½，因此需要再錄音改變音

量的大小。同時，因爲APPLE沒有 SYM使用的高頻電容器，所以音調的控制也要調整。SYNC 磁帶能使你適當的控制你的錄音機（請看 SYM使用手册的附錄F）。要做一個SYNC磁帶，首先將SYMOUT程式載入你的APPLE，接着設立一些參數以決定那一種形式，MODE（位址$11E0）設爲$80表爲SYM格式，設爲$00表爲KIM 格式，並且從程式的SYNC（$1000）開始執行。這是一個無止盡的廻路（loop），所以在按RESET之前要把輸出錄個幾分鐘，再來利用此錄好的卡帶，讀入錄音機，把錄音機的音量與音調調好。

一旦一切都調好後，要將你的程式載入就非常簡單了。首先載入SYMOUT程式。然後再將你要執行的程式載入到RAM中。接着填入下列參數：起始位址（$11DB－C），結束位址（$11DD－E），磁帶編號（$11DF），以及MODE（$11E0），並從程式的SYMOUT（$1080）開始執行。錄下此程式，存入你的 SYM，你便可以擁有它了!

電腦和電腦直接通信

Dr. Bohanan 發現：假如你的錄音機有監聽接點，即你可以聽到所錄的聲音，你可以利用它，將APPLE直接和SYM 連接，如此便可以減少發生錯誤的可能性。在我們的SYM上，若用SONY錄音機，則載入15D0個位元組的程

式，其成功率爲 70 %。錄音機的音量和音調有很大的影響。當二部電腦直接經由錄音機的監聽接點連接起來時，我們測試時每次都可載入成功，並且音量與音調沒有影響。我發現我的許多錄音機依此方式來做效果都不錯，且經由耳機的接點可以達到監聽的效果。

```
0800              ;************************
0800              ;*                      *
0800              ;*    SYM-KIM FORMAT    *
0800              ;*   CASSETTE  OUTPUT   *
0800              ;*       S.WELCH        *
0800              ;*                      *
0800              ;*       SYM-KIM        *
0800              ;*                      *
0800              ;*  COPYRIGHT (C) 1981  *
0800              ;*    MICRO INK, INC.   *
0800              ;* CHELMSFORD, MA 01824 *
0800              ;*  ALL RIGHTS RESERVED *
0800              ;*                      *
0800              ;************************
0800              ;
0800              ;
0800              ;LARGELY COPIED FROM THE
0800              ;SYNERTEX MANUAL, AND RE-
0800              ;PRODUCED HERE WITH THE
0800              ;PERMISSION OF SYNERTEX
0800              ;SYSTEMS CORP.
0800              ;
0800              ;
0800              TAPOUT EQU $C020
0800              ;
0800              ;USE APPLE GAME PADDLE ANNUNCIATOR #0 FOR TAPE RECORDER
0800              ;ON-OFF CONTROL. RECORDER ON IS LOW.
0800              ;
0800              TAPEON EQU $C059             ;PUT 0 HERE TO TURN ON
0800              TAPEOF EQU $C058             ;PUT 1 HERE TO TURN OFF
0800              TM1500 EPZ $47               ;PROB SHOULD BE TWEAKED
0800              TIME99 EPZ $1A               ;FOR DELAY ROUTINE
0800              EOT    EPZ $04
0800              SYN    EPZ $16
0800              BUFADL EPZ $E7               ;ARBITRARY PLACE ON ZERO PAGE
0800              BUFADH EPZ $E8
0800              CHAR   EPZ $EA
0800              ;
0800              ;
0800              ;---PROGRAM STARTS HERE, LINE 390 OF SYM CODE LOC 8E87
0800              ;
0800              BEGIN  EQU $1080             ;MUST START IN MIDDLE OF PAGE
1080                     ORG BEGIN             ;OUT OF WAY OF MOST SYM PROGRAMS
1080                     OBJ $880
1080              ;
1080              ;--INITIALIZE--
1080 20BB11       SYMOUT JSR START             ;ENTRY-PARAMETERS SET BEFORE CALL
1083 A080                LDY #$80              ;INCASE WE TAKE KIM BRANCH
```

```
1085 2CE011          BIT MODE           ;TEST BIT 7 OF MODE (1=SYM,0=KIM)
1088 100D            BPL DUMPT1         ;KIM-DO 128 SYNS
108A         ;
108A         ;--WRITE 8 SECOND MARK--
108A A208            LDX #$8            ;8 TIMES...
108C A015    MARK8A  LDY #$15           ;ONE SEC (21 DELAYSPER SEC)
108E 209511  MARK8B  JSR DELAY          ;BENIGN PAUSE, SYM USES KIM CHAR
1091 88              DEY
1092 D0FA            BNE MARK8B
1094 CA              DEX
1095 D0F5            BNE MARK8A
1097         ;--WRITE 256 SYNS FOR SYNC--
1097 A916    DUMPT1  LDA #SYN
1099 200711          JSR OUTCTX
109C 88              DEY
109D D0F8            BNE DUMPT1
109F         ;--WRITE START CHARACTER--
109F A92A            LDA #'*'
10A1 200711          JSR OUTCTX
10A4         ;--WRITE ID--
10A4 ADDF11          LDA ID
10A7 203B11          JSR OUTBTX
10AA         ;---WRITE STARTING ADDRESS---
10AA ADDB11          LDA SAL
10AD 203811          JSR OUTBCX
10B0 ADDC11          LDA SAH
10B3 203811          JSR OUTBCX
10B6 2CE011          BIT MODE           ;KIM OR HS?
10B9 100C            BPL DUMPT2
10BB         ;---WRITE ENDING ADDRESS---
10BB ADDD11          LDA EAL
10BE 203811          JSR OUTBCX
10C1 ADDE11          LDA EAH
10C4 203811          JSR OUTBCX
10C7         ;---START OF MEMORY DUMP---
10C7         ;--FIRST CHECK IF THIS IS THE LAST BYTE OUT---
10C7 A5E7    DUMPT2  LDA BUFADL         ;LOAD ADDRESS OF CURRENT BYTE
10C9 CDDD11          CMP EAL
10CC D029            BNE DUMPT4         ;COMPARE TO ENDING ADDRESS
10CE A5E8            LDA BUFADH
10D0 CDDE11          CMP EAH
10D3 D022            BNE DUMPT4         ;BRANCH IF MORE TO OUTPUT
10D5 A92F            LDA #'/'           ;YUP, LAST BYTE: WRITE '/'
10D7 200711          JSR OUTCTX
10DA         ;---WRITE CHECKSUM---
10DA ADE111          LDA CHKL
10DD 203B11          JSR OUTBTX
10E0 ADE211          LDA CHKH
10E3 203B11          JSR OUTBTX
10E6         ;---WRITE TWO EOT'S---
10E6 A904            LDA #EOT
10E8 203B11          JSR OUTBTX
10EB A904            LDA #EOT
10ED 203B11          JSR OUTBTX
10F0         ;---OK, NOW WE'RE ALL DONE, SO CLEAN UP AND EXIT---
10F0 18              CLC                ;INDICATE SUCCESS
10F1         ;---SKIPPED LOTS OF STUFF, MOSTLY SYM SPECIFIC---
10F1 A201            LDX #$01           ;SHUT OFF TAPE RECORDER
10F3 8E58C0          STX TAPEOF
10F6 60              RTS                ;AND WE'RE ALL DONE
10F7         ;NEXT IS THE CODE WHICH OUTPUTS THENEXT MEM LOCATION
10F7 A000    DUMPT4  LDY #$0            ;FIND THE NEXT BYTE
10F9 B1E7            LDA (BUFADL),Y
10FB 203811          JSR OUTBCX         ;WRITE IT AND UPDATE CHCKSUM
```

```
10FE E6E7              INC BUFADL           ;BUMP BUFFER ADDR
1100 D0C5              BNE DUMPT2
1102 E6E8              INC BUFADH           ;CARRY
1104 4CC710            JMP DUMPT2           ;GO BACK AND SEE IF WE'RE DONE
1107            ;
1107            ;
1107            ;START OF VARIOUS CHARACTER OUT ROUTINES
1107            ;
1107 2CE011     OUTCTX BIT MODE             ;HS OR KIM?
110A 1047              BPL OUTCHT           ;KIM TAKES BRANCH
110C            ;OUTBTH - NO CLOCK A,X DESTROYED
110C            ;MUST RESIDE ON ONE PAGE - TIMING CRITICAL
110C            ;
110C A209       OUTBTH LDX #$9              ;8 BITS+START BIT
110E 8CE411            STY TEMP2
1111 85EA              STA CHAR
1113            ;CAN'T READ LEVEL ON APPLE, SO NEXT INSTRUCTION IS DUMMY
1113 ADE311            LDA TEMP1            ;FOR TIMING
1116 46EA       GETBIT LSR CHAR
1118 49E5              EOR #TPBIT
111A 8D20C0            STA TAPOUT           ;INVERT LEVEL
111D            ;HERE STARTS FIRST 416 USEC PERIOD
111D A047              LDY #TM1500
111F 88         A416   DEY                  ;TIME FOR THIS LOOP IS 5Y-1
1120 D0FD              BNE A416
1122 9011              BCC NOFLIP           ;NOFLIP IF BIT 0
1124 49E5              EOR #TPBIT           ;BIT IS 1 - INVERT OUTPUT
1126 8D20C0            STA TAPOUT           ;END OF FIRST 416 USEC PERIOD
1129 A046       B416   LDY #TM1500-1
112B 88         B416B  DEY                  ;LENGTH OF LOOP IS 5Y-1
112C D0FD              BNE B416B
112E CA                DEX
112F D0E5              BNE GETBIT           ;GET NEX BIT (LAST IS 0START BIT
1131 ACE411            LDY TEMP2            ;(BY 9 BIT LSR)
1134 60                RTS
1135 EA         NOFLIP NOP                  ;TIMING
1136 90F1              BCC B416             ;(ALWAYS)
1138            ;
1138 20AC11     OUTBCX JSR CHKT             ;GO UPDATE CHECKSUM
113B 2CE011     OUTBTX BIT MODE
113E 08CC              BMI OUTBTH           ;HS
1140            ;OUTBTC - OUTPUT ONE KIM BYTE
1140 A8         OUTBTC TAY                  ;SAVE DATA BYTE
1141 4A                LSR
1142 4A                LSR
1143 4A                LSR
1144 4A                LSR                  ;SHIFT HI NIBBLE INTO PLACE
1145 204811            JSR HEXOUT           ;AND OUTPUT HI NIBBLE FIRST
1148 290F       HEXOUT AND #$0F             ;CONVERT LO NIBBLE TO ASCII
114A C90A              CMP #$0A
114C 18                CLC
114D 3002              BMI HEX1
114F 6907              ADC #$07
1151 6930       HEX1   ADC #$30
1153            ;
1153            ;OUTCHT OUTPUTS AND ASCII CHAR IN KIM FORMAT
1153            ; (MUST RESIDE ON ONE PAGE, FOR TIMING)
1153            ;
1153 8EE311     OUTCHT STX TEMP1            ;SAVE X & Y
1156 8CE411            STY TEMP2
1159 85EA              STA CHAR
115B A9FF              LDA #$FF             ; USE FF W/SHIFTS TO COUNT BITS
115D 48         KIMBIT PHA                  ;SAVE BIT COUNTER
```

```
115E ADE411   176           LDA TEMP2          ;DUMMY FOR TIMING
1161 46EA     177           LSR CHAR           ;GET DATA BIT IN CARRY
1163 A212     178           LDX #$12           ; ASSUME ONE
1165 B002     179           BCS HF
1167 A224     180           LDX #$24           ;BIT IS ZERO
1169 A019     181   HF      LDY #$19
116B 49E5     182           EOR #TPBIT         ;DUMMY, REALLY
116D 8D20C0   183           STA TAPOUT         ;INVERT OUTPUT BIT
1170 88       184   HFP1    DEY                ;PAUSE FOR 138 USEC
1171 D0FD     185           BNE HFP1
1173 CA       186           DEX
1174 D0F3     187           BNE HF
1176 A218     188   LF      LDX #$18           ;ASSUME BIT IS ONE
1178 B002     189           BCS LF20
117A A20C     190           LDX #$0C           ;BIT IS ZERO
117C A027     191   LF20    LDY #$27
117E 49E5     192           EOR #TPBIT         ;DUMMY
1180 8D20C0   193           STA TAPOUT         ;INVERT OUTPUT
1183 88       194   LFP1    DEY                ;PAUSE FOR 208 USEC
1184 D0FD     195           BNE LFP1
1186 CA       196           DEX
1187 D0F3     197           BNE LF20
1189 68       198           PLA                ;RESTORE BIT CTR
118A 0A       199           ASL                ;DECREMENT IT
118B D0D0     200           BNE KIMBIT         ;FF SHIFTED 8X-00
118D AEE311   201           LDX TEMP1
1190 ACE411   202           LDY TEMP2
1193 98       203           TYA                ;RESTORE X,Y, DATA BYTE
1194 60       204           RTS
1195          205   ;WE NEED A DELAY FUNCTION, BECAUSE THE SYM PROG
1195          206   ;USES THE KIM CHARGOUT ROUTINE WITH OUT PUT DISABLED
1195          207   ;TO DELAY (AND WE CAN'T)
1195          208   ;
1195          209   ;THIS ONE SHOULD BE 1/21 SECOND, SINCE IT EMULATES
1195          210   ;THE KIM CHAR OUT ROUTINE, WHICH THE SYM PROGRAM USES
1195 8EE311   211   DELAY   STX TEMP1          ;PRESERVE X
1198 8CE411   212           STY TEMP2          ;AND Y
119B A200     213           LDX #$00           ;DO OUTER LOOP 256 TIMES
119D A01A     214   LOOP0   LDY #TIME99        ;LOOP
119F 88       215   LOOP1   DEY
11A0 D0FD     216           BNE LOOP1
11A2 CA       217           DEX
11A3 D0F8     218           BNE LOOP0
11A5 AEE311   219           LDX TEMP1          ;RESTORE X
11A8 ACE411   220           LDY TEMP2          ;AND Y
11AB 60       221           RTS
11AC          222   ;
11AC          223   ;CHKT...UPDATE CHECKSUM FROM BYTE IN ACC
11AC A8       224   CHKT    TAY                ;SAVE ACC
11AD 18       225           CLC
11AE 6DE111   226           ADC CHKL
11B1 8DE111   227           STA CHKL
11B4 9003     228           BCC CHKT10
11B6 EEE211   229           INC CHKH           ;BUMP HI BYTE
11B9 98       230   CHKT10  TYA                ;RESTORE ACC
11BA 60       231           RTS
11BB          232   ;START---LEAVING OUT SOME UNECESSARY JUNK
11BB 20C711   233   START   JSR ZERCK          ;ZERO CHECKSUM
11BE 20D011   234           JSR P2SCR          ;THATS WHAT THEY NAMED IT
11C1 A900     235           LDA #$00           ;TURN ON TAPE RECORDER
11C3 8D59C0   236           STA TAPEON
11C6 60       237           RTS
11C7 A900     238   ZERCK   LDA #$00           ;ZERO CHECKSUM
```

```
11C9 8DE111          STA CHKL
11CC 8DE211          STA CHKH
11CF 60              RTS
11D0         ;--P2SCR-- THIS MOVES THE STARTING ADDRESS
11DC         ;          TO THE RUNNING BUFFER ADDRESS.
11D0         ;THE WEIRD NAME IS DUE TO THE NAMES
11D0         ;OF THE LOCATIONS WHICH WE ARE MOVING IN THE SYM BOOK
11D0 ADDC11  P2SCR   LDA SAH             ;STARTING ADD HI
11D3 85E8            STA BUFADH
11C5 ADDB11          LDA SAL             ;STARTING ADD LO
11D8 85E7            STA BUFADL
11DA 60              RTS
11DB         ;PAGE PARAMETERS, ETC.
11DB         ;THESE NEXT SIX LOCATIONS SHOULD BE
11DB         ;FILLED WITH THE CALLING PARAMETERS
11DB         ;BEFORE CALLING THE SYMOUT ROUTINE
11DB         ;
11DB         ;
11DB 00      SAL     HEX 00              ;STARTING ADDRESS, LO BYTE
11DC 00      SAH     HEX 00              ;STARTING ADDRESS, HI BYTE
11DD 00      EAL     HEX 00              ;ENDING ADDRESS+1, LO BYTE
11DE 00      EAH     HEX 00              ;ENDING ADDRESS+1, HI BYTE
11DF 00      ID      HEX 00              ;TAPE ID NUMBER
11E0 00      MODE    HEX 00              ;SYM=$80, KIM=$00
11E1 00      CHKL    HEX 00              ;VARIABLES
11E2 00      CHKH    HEX 00
11E3 00      TEMP1   HEX 00
11E4 00      TEMP2   HEX 00
11E5 00      TPBIT   HEX 00
11E6         ;
11E6         ;--- SHORT ROUTINE TO MAKE SYNC TAPES
11E6         ;     (APPLE PRODUCED TAPE WILL USUALLY NEED
11E6         ;      DIFFERENT VOLUME AND TONE SETTINGS
11E6         ;      THAN KIM OR SYM TAPES)
11E6         ;
1000                 ORG $1000
1000                 OBJ $800
1000         ;
1000 20BB11  SYNC    JSR START           ;MAKE A SYNC TAPE
1003 A916    SYNMOR  LDA #SYN            ;LOAD SYNC CHARACTER
1005 200711          JSR OUTCTX          ;SEND IT
1008 4C0310          JMP SYNMOR          ;DO IT FOREVER
100B         ;
                     END
```

```
***************************
*                         *
* SYMBOL TABLE -- V 1.5   *
*                         *
***************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

TM1500 0047  TIME99 001A  EOT    0004  SYN    0016  BUFADL 00E7  BUFADH 00E8
CHAR   00EA
```

```
** ABSOLUTE VARABLES/LABELS

TAPOUT C020   TAPEON C059   TAPEOF C058   BEGIN  1080   SYMOUT 1080
MARK8A 108C   MARK8B 108E   DUMPT1 1097   DUMPT2 10C7   DUMPT4 10F7   OUTCTX 1107
OUTBTH 110C   GETBIT 1116   A416   111F   B416   1129   B416B  112B   NOFLIP 1135
OUTBCX 1138   OUTBTX 113B   OUTBTC 1140   HEXOUT 1148   HEX1   1151   OUTCHT 1153
KIMBIT 115D   HF     1169   HFP1   1170   LF     1176   LF20   117C   LFP1   1183
DELAY  1195   LOOP0  119D   LOOP1  119F   CHKT   11AC   CHKT10 11B9   START  11BB
ZERCK  11C7   P2SCR  11D0   SAL    11DB   SAH    11DC   EAL    11DD   EAH    11DE
ID     11DF   MODE   11E0   CHKL   11E1   CHKH   11E2   TEMP1  11E3   TEMP2  11E4
TPBIT  11E5   SYNC   1000   SYNMOR 1003

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:01DA
```

6 參考資料

簡介

從整數 BASIC 截取 DOS 錯誤訊號
——— Andy Hertzfeld

APPLESOFT 的浮點常規
——— R. M. Mottola

如何使用傳送口 (hooks)
——— Richard Williams

BROWN 和 WHITE 與所有的彩色
——— Richard F. Suitor

第六部份簡介

本章將爲那些想多知道一些APPLE的韌體(firmware)與硬體功能的使用者提供一些參考資料。每一篇都探討APPLE不同的功能。

Andy Hertzfeld的“從整數BASIC截取DOS的錯誤訊號”。提供DOS錯誤訊號碼，並指出如何從整數BASIC截取它們。R.M. Mottola的“APPLESOFT的浮點常規”，討論存在APPLESOFT韌體內的一些有用的浮點常規。這些常規在MEAN-14系統中有使用到(看第一部份)。Richard Williams的“如何使用傳送口(hooks)”，說明監督系統傳送口的使用法，並說明如何使用它們取得控制權。並舉了二個例題。最後，Dick Suitor的“BROWN和WHITE與所有的彩色”，討論APPLE彩色圖形的一些理論，並提供一些範例程式。

使用這些程式之前，你必須對你的APPLE要有充分的了解，以及你的APPLE有些什麼。第三篇有關傳送口的文章，可幫助初學者了解第一部份的常規。

從整數BASIC截取DOS錯誤訊號

作者：Andy Hertzfeld

此整數BASIC程式能從DOS取得錯誤訊號，診斷是何種錯誤，然後採取修正動作，避免操作者介入。

當在執行一個BASIC程式時，DOS發出FILE NOT FOUND的錯誤訊號，則程式會暫時停止，且會印出錯誤消息。不幸的，這並不是我們所期望發生的。我們希望程式在被告知錯誤後能繼續執行，並依其所想要的任何方法修正。

這種技巧很容易做，因爲在APPLESOFT中有ONERR錯誤截取的功能。從整數BASIC則比較困難；本章將提供一種方法來做。

DOS不像整數BASIC，DOS是存在RAM 中。這表示我們能利用它做一些我們想做的事。位址9D5A（48K的系統）爲BASIC錯誤處理常規的位址，當錯誤發生時DOS便會到此來執行。對於整數BASIC通常含有E3E3，ROM APPLESOFT爲D865。因此，我們可以將我們的常規位址存在9D5A。（32K的系統爲5D5A），所以當

發生DOS錯誤時我們就可取得控制權。

下列的 24 個位元組，是可以重新定位的常規，它可以從BASIC截取錯誤。當發生 DOS 錯誤時，位址 2 將儲存錯誤碼；發生錯誤的行數號碼會存在位址 3 和 4 ；最後，將控制權傳到BASIC 指述，此指述的行號存在位址 0 和 1 。因爲此常規是可以重新定位的，所以可以放在任何位址。位址 300 是一個很好的位址，除非你想將列表機驅動程式放在此位址上。

爲了啓動此錯誤截取常規，執行下列二個POKE，它將把截取常規的位址存在 $9D5A :

POKE- 25254, 0: POKE- 25253, 3 (48K 系統)或是

POKE- 23898, 0: POKE- 23899, 3 (32K 系統)。

此錯誤截取常規本身可以POKE到第 3 頁，或是BLOAD到磁碟，隨你喜歡。假如不是定位在 $300 ，要相對的改變上述POKE 指令。

當此常規載入到記憶體後，就很容易使用了。假如你的錯誤處理常規是由行號LINE 開始，則你應該利用下面二個指令，“ POKE 0, LINE mod 256 ”與“ POKE 1, LINE/ 256 ”，告訴錯誤截取常規你要跳到那裏。你的BASIC 處理常規能把位址 3 和 4 PEEK出來，便可知道是那一行發生錯誤。

PEEK(3)+ 256 * PEEK(4) 就是行號。把位址 2 的值PEEK出來便可知道發生那一種錯誤。表 1 列出各種不同錯誤的號碼。

不幸的，在此仍有一個小問題。縱使在發生錯誤時你可以取得控制權，但是 DOS 仍會發出響聲並印出任何錯誤消息。只要一個POKE便可禁止DOS 做這些事情，但是這個 POKE將會抑制所有DOS 的錯誤消息，包括立即執行的錯誤，所以有一點不好。而且對於不同大小的系統與不同的 DOS有不同的POKE指令。

48K, DOS V3.1 :	POKE - 22978, 20
48K, DOS V3.2/3.3 :	POKE - 22820, 18
32K, DOS V3.1 :	POKE - 26174, 20
32K, DOS V3.2/3.3 :	POKE - 26332, 18

表1——錯誤號碼與消息

號碼	消息
1	Language Not Available
2	Range Error
3	Range Error
4	Write Protection Error
5	End of Data Error
6	File Not Found Error
7	Volume Mismatch Error
8	Disk I/O Error
9	Disk Full Error
10	File Locked Error
11	Syntax Error
12	No Buffers Left Error
13	File Type Mismatch
14	Program Too Large Error
15	Not Direct Command

注意，這些是DOS V3.2和V3.3的錯誤消息；V3.1的錯誤消息有一點不同。

在整個系統，你可以把4 POKE到上面所舉例的位址上，便可恢復錯誤消息的功能。

捕獲 DOS 錯誤消息的能力是非常重要的，特別是在一些不允許程式暫停執行的重要系統上。也許各位可以利用整數 BASIC 寫出比此常規更好，更快的程式。

```
0800           ;************************
0800           ;*                      *
0800           ;*     INTERCEPTING     *
0800           ;*      DOS ERRORS      *
0800           ;*   BY ANDY HERTZFELD  *
0800           ;*                      *
0800           ;*        ERROR         *
0800           ;*                      *
0800           ;*  COPYRIGHT (C) 1981  *
0800           ;*    MICRO INK, INC.   *
0800           ;* CHEMSFORD, MA 01824  *
0800           ;* ALL RIGHTS RESERVED  *
0800           ;*                      *
0800           ;************************
0800           ;
0800           ;
0800           ERNUM   EPZ $02             ;ERROR NUMBER
0800           ERRLIN  EPZ $03             ;LINE OF ERROR
0800           ONERR   EPZ $00             ;CONTROL TRANSFER LINE
0800           ;
0800           PR      EPZ $DC             ;BASIC LINE POINTER
0800           ACL     EPZ $CE             ;BASIC ACCUMULATOR
0800           ACH     EPZ $CF
0800           ;
0800           GOTO    EQU $E85E           ;BASIC 'GOTO' ROUTINE
0800           ;
0300                   ORG $300
0300                   OBJ $800
0300           ;
0300           ;
0300 8602      ERROR   STX ERNUM           ;SAVE ERROR NUMBER
0302 A001              LDY #01
0304 B1DC              LDA (PR),Y          ;GET LOW BYTE OF ERRING
0306 8503              STA ERRLIN          ;LINE NUMBER AND SAVE
0308 C8                INY
0309 B1DC              LDA (PR),Y          ;DITTO FOR HIGH BYTE
030B 8504              STA ERRLIN+1
030D A500              LDA ONERR           ;GET LOW BYTE OF LINE NUMBER
030F 85CE              STA ACL             ;OF ERROR HANDLING STATEMENT
```

```
0311 A501          LDA CNERR+1          ;DITTO FOR HIGH BYTE
0313 85CF          STA ACH              ;SET THINGS UP FOR BASIC AND
0315 4C5E          JMP GOTO             ;LET THE FIRMWARE TAKE OVER
                  END

        **************************
        *                        *
        * SYMBOL TABLE -- V 1.5  *
        *                        *
        **************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

ERNUM  0002  ERRLIN 0003  CNERR  0000  PR     00DC  ACL    00CE  ACH    00CF

** ABSOLUTE VARABLES/LABELS

GOTO   E85E  ERROR  0300

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0052
```

APPLESOFT的浮點常規

作者：R. M. Mottola

APPLESOFT BASIC是一個非常容易使用的語言——但是它有時慢的令人厭煩。爲了減少執行的時間，有一些程式設計師因此把某些常規用機器語言寫。然而有些常規已經存在APPLESOFT 翻譯器中，所以不必再浪費時間重寫。解決的方法是：直接使用浮點常規！在此我們將討論浮點常規在那裏，它們有什麼作用，以及直接使用它們的範例。

在最近的一項計劃中，需要我寫一個常規計算各種統計資料。開始用APPLESOFT浮點BASIC 寫，效果不錯只不過執行時間太長。做過一些試驗後，我們知道有許多時間都花在轉換BASIC的資料型式，浮點的“FOR-NEXT”廻路，與一些和翻譯器有關的函數。

我所需要的是將所有的常規都用機器語言寫。要如此做，則有二種選擇。第一，寫一些浮點常規，使其有APPLESOFT的5個位元組的變數格式。但這不合乎實際，因爲所需要的記憶體太多了。

第二，比較有效率的方法是直接使用 APPLESOFT 中的浮點常規。由許多理由得知這種方法非常難，但我經過長久的探討終於發現了下列一些常規。在使用它們之前，你最好先熟悉APPLESOFT變數的格式與APPLESOFT 浮點累加器的格式。

APPLESOFT 變數的格式是一個標準的 5 個位元組浮點表示式，最高位元組含有指數（ exponent ），其餘 4 個位元組含有正負值的假數。（看APPLESOFT 使用手册第 137 頁）APPLESOFT累加器的格式有一點不同。你從各種APPLESOFT 第零頁使用表中便可注意到二個浮點累加器都各分配有 7 個位元組。這些累加器的格式如下：最高位元組含有指數。下四個位元組含有假數的負絕對值，如 APPLESOFT 變數格式表示法。假如某值由變數格式轉換成累加器格式，則第 6 個位元組含有此值原來假數的最高位元組。在任何情況下，此位元組是用來表示假數的正負號。第 7 個位元組是一個“ 函數 ”位元組，用於算術運算。將某值由變數格式轉換成累加器中格式時，此位元組不會有初值存在。

使用下列的浮點常規是一件非常容易的事情。為了簡單起見，您可以利用下面所列的“ FPLOAD ”副常規將所有的值都載入到累加器，而不必記住某值在累加器中格式如何。此常規在載入時會執行轉換動作。將所有的值都表示成正規形式（ normalized form ）時要小心一點。假如您只要使用APPLE-SOFT已經定義的值，則就不必再做此動作，因為APPLESOFT

將所有定義的值都正規化。為了要使用您自己定義的值，您可能會發現所附的共用程式非常有用。

另外一件要注意的事情是有關浮點錯誤（除以零，滿溢現象（ overflow ））。因為這些浮點常規並不會在APPLE-SOFT之外使用，錯誤處理常規的進入點是在 ROM 中。不幸的，指向此常規的向量（ vectors ）是固定的，不能改變。在此有二種方法處理這種錯誤：

1. 用“最壞的情況”測試你的常規。假如你能保證不會發生錯誤，那就可以了。
2. 可以利用APPLESOFT的ONERR…… GOTO 指述，使得發生錯誤時，會跳到指定的BASIC 行數。在此行數中，你可以呼叫自己的機器語言錯誤處理常規。

下列的常規構成APPLESOFT 中有效的主要算術運算常規。當然，還有一些其它函數沒有在此提出。

名稱：FPLOAD

位址：$EAF9

符號：M→ FPAC1

將變數載入主浮點累加器。轉換成FPAC格式。A和Y暫存器必須指向記憶體中的變數（ ADL, ADH ）。清除 $AC。

名稱： FPSTR

位址： $EB2B

符號： FPAC1→M

將主浮點累加器的值存入記憶體中。將FPAC格式轉換成APPLESOFT 變數格式。X和Y暫存器必須指向此值在記憶體中所存的（ADL, ADH）第一個位元組。清除$AC。

名稱： TR1 > 2

位址： $EB63

符號： FPAC1 → FPAC2

將存在主浮點累加器的值傳送到次浮點累加器。清除$AC。

名稱： FPDIV2

位址： $EA60

符號： FPAC2/M→FPAC1

將以浮點累加器中的值除以暫存器A和Y所指的值（ADL, ADH），並且把結果存在主浮點累加器。

名稱： TR2 > 1

位址： $EB53

符號： FPAC2 → FPAC1

將存在次浮點累加器的值傳送到主浮點累加器，清除$AC。

名稱：FPSQR
位址：$EE8D
符號：FPAC1→FPAC1

取主浮點累加器的正開根號值，並存回主浮點累加器。

名稱：FPEXP
位址：$EE94
符號：FPAC2↑M→FPAC1

把A和Y暫存器所指的值當作指數，次浮點累加器的值當作假數，並將乘積結果存於主浮點累加器。

名稱：FPINT
位址：$EC23
符號：INT（FPAC1）→FPAC1

取主浮點累加器中的整數值，並存於主浮點累加器。

名稱：FPABS
位址：$EBAF
符號：ABS（FPAC1）→FPAC1

取主浮點累加器中的絕對值，並存於主浮點累加器。

名稱：FPADD

位址：$E7BE

符號：M+FPAC1→FPAC1

將暫存器A和Y（ADL, ADH ）所指的值加上主浮點累加器的值，並將結果存於主浮點累加器。

名稱：FPADD2

位址：$E7A0

符號：0.5+FPAC1→FPAC1

和前一個常規類似，在此是把主浮點累加器加上0.5。

名稱：FPMOL

位址：$E97F

符號：M*FPAC1→FPAC1

將暫存器A和Y（ADL, ADH ）所指的值乘上主浮點累加器的值，並將結果存於主浮點累加器。

名稱：FPSUB

位址：$E7A7

符號：M－FPAC1→FPAC1

把暫存器A和Y（ADL, ADH ）所指的值減掉主浮點累加器的值，並將結果存於主浮點累加器。

名稱：FPDIV

位址：$EA66

符號：M/FPAC1→FPAC1

把暫存器A和Y（ADL, ADH ）所指的值除以主浮點累加器的值，並將結果存於主浮點累加器。

名稱：FPSGN

位址：$EB90

符號：SGN（FPAC1）→FPAC1

將主浮點累加器的值的正負號存入主浮點累加器。負數就存入（－1），正數存入（1），零則存入（0）。

名稱：FPLOG

位址：$E941

符號：LOG（FPAC1）→FPAC1

取主浮點累加器的自然對數，並把結果存入主浮點累加器。

名稱： COMP2
位址： $E89E
符號： 取FPAC1的2位補數→FPAC1

取主浮點累加器的2位補數，並把結果存入主浮點累加器。

名稱： INT＞FP
位址： $E2F2
符號： (Y, A)→FPAC1

將一個二個位元組的整數轉換成相對的浮點值（FPAC格式），並將其存入主浮點累加器。此整數的高位元組要存在暫存器A中，低位元組要存在暫存器Y中。

名稱： FP＞INT
位址： $E10C
符號： FPAC1→($A0, $A1)

將主浮點累加器中的浮點數值轉換成二個位元組的整數，並將其存在主浮點累加器的第4與第5個位元組（$A0, $A1）。$A0存着高位元組，$A1存着低位元組。

```
1   REM ************************
2   REM *                      *
3   REM *    FLOATING POINT    *
4   REM *       ROUTINES       *
5   REM *     R.M. MOTTOLA     *
6   REM *                      *
7   REM *  COPYRIGHT (C) 1981  *
8   REM *    MICRO INK, INC.   *
9   REM * CHELMSFORD, MA 01824 *
10  REM *  ALL RIGHTS RESERVED *
11  REM *                      *
12  REM ************************
13  REM
14  REM
80 :
90 X = 0:D$ =  CHR$ (4)
100  FOR N = 768 TO 792
110  READ A: POKE N,A
120  NEXT
130  REM  ESTABLISH CONVERSION ROUTINE AT $300
140  DATA  165,105,24,105,2
150  DATA  164,106,144,1,200
160  DATA  32,249,234,160,6
170  DATA  185,157,0,153,25
180  DATA  3,136,16,247,96
190  HOME : PRINT : PRINT  TAB( 7)"FLOATING POINT CONVERSIONS"
200  PRINT : PRINT : PRINT "INSTRUCTIONS-"
210  PRINT : PRINT "ENTER VALUE YOU WISH CONVERTED TO FLOATING POINT
    REPR  ESENTATION.  IF YOU WISH TO PRINT THE CONVERSIONS ON THE"
220  PRINT "PRINTER, FOLLOW THE VALUE WITH A 'P'.  TO RETURN TO BASIC,
     HI    T (RETURN) KEY."
230  VTAB 14: CALL  - 868
240  INPUT "ENTER VALUE: ";A$
250  IF A$ = "" THEN  VTAB 23: END : REM  ""=NULL$
260  IF  RIGHT$ (A$,1) >  < "P" THEN 300
270  PRINT D$;"PR#1"
280  REM  PRINTER IN SLOT #1
290  PRINT : PRINT
300 X =  VAL (A$): CALL 768
310  VTAB 18: CALL  - 958: PRINT "VALUE= "X
320  PRINT : PRINT "ACCUMULATOR: $";
330  FOR N = 793 TO 799
340 A =  PEEK (N): GOSUB 450
350  NEXT : PRINT : PRINT
360  PRINT "VARIABLE:    $";
370 B =  PEEK (105) +  PEEK (106) * 256 + 2
380  FOR N = B TO B + 4
390 A =  PEEK (N): GOSUB 450
400  NEXT : PRINT
410  PRINT D$;"PR#0"
420  GOTO 230
430 :
440  REM  DECIMAL TO HEX SUB
450 A = A / 16:B =  INT (A)
460 A = (A - B) * 16
470 B = B + 48: IF B > 57 THEN B = B + 7
480  PRINT  CHR$ (B);
490 A = A + 48: IF A > 57 THEN A = A + 7
500  PRINT  CHR$ (A)" ";
510  RETURN
```

如何使用傳送口(hooks)

作者： Richard Williams

一旦你知道如何使用有效的傳送口後，你對你的APPLE便有很多事可以做。

APPLE II 允許你使用你自己的輸入與輸出常規來取代APPLE II 的標準常規。圖1顯示當一個字元由APPLE II 輸出時的基本控制流程。圖2顯示當你用你自己的輸出常規取代標準監督系統常規時控制流程的改變。它所使用的就是“傳送口”(hooks)的概念，你可以中斷正常的控制流程而把控制權傳到你自己的常規。

圖3顯示使用此方法的範例。平常控制字元並不會顯示在螢幕上。然而，可加一個常規，在印出時將控制字元變成反相視頻訊號，字元便會顯示在螢幕上。此程序在列印含有控制字元的程式時很有用。

它如何工作

在眞正做輸入與輸出之前，系統將會經由第 0 頁記憶體間接的跳到眞正的輸入與輸出常規。只要改變跳越的位址，你便可以用你自己的常規取代標準常規。在輸入方面，在監督程式位址 $FD18 有一個 JMP（KSWL）指令。KSWL（$38）和 KSWH（$39）含有輸入常規的位址。同樣的，在輸出方面，在位址 $FDED 上有一個 JMP（CSWL）指令，它將跳到輸出常規。CSWL（$36），與 CSWH（$37）含有輸出常規的位址。這些資料可在 APPLE II 參考手册第 166 頁和第 167頁查到。

如何加入一個輸入常規

正常的輸入常規是 KEYIN，在位址 $FD1B。若要換成你的常規，則將你的常規的位址存入 KSWL 和 KSWH。你的輸入常規需要做下列工作：

1. 進入你的常規後，累加器將會含有由提示符號（prompt）取代的字元。你必須用一個 STA（BASL），Y，其中 BASL = $28，把此字元回復到螢幕上。要在改變暫存器 A 或 Y 之前做好。

2. 假如字元是來自鍵盤，就清除鍵盤激勵訊號（ strobe ）。
3. 將在累加器中的字元送回，且要設立字元的高位元為 1。
4. 正常輸入常規在等待輸入時，增加隨機數字的基值（ seed ）。所以在你的常規中也要做這件工作。

假如你希望從鍵盤輸入，則只要呼叫KEYIN（ JSR $FD1B ）便可做完上述各項工作。你可以任意的處理此輸入字元，它是存在累加器中，然後用一個 RTS 返回。假如你想用你自己的常規取代KEYIN，則首先要把KEYIN弄清楚。

圖 1	圖 2	圖 3
程式的輸出部份	程式的輸出部份	程式的輸出部份
↓	↓	↓
監督程式的輸出處理部門	監督程式的輸出處理部門	監督程式的輸出處理部門
↓	↓	↓
監督程式的字元輸出常規	使用者的字元輸出常規	將控制字元轉換成反相視頻訊號的使用者常規
		↓
		監督程式的字元輸出常規

如何加入一個輸出常規

正常的輸出常規COUT1 在位址 $FDF0。要加入你的常規，要把你的常規位址存入CSWL 和CSWH（ $36 和$37 ）。在你的常規被呼叫之前，要輸出的字元將會放在累加器中。假如你希望在你的常規做完之前，把存在累加器中的字元印在螢幕上，則可在你的常規結束之前做一個JMP COUT1 。將控制字元轉換成反相視頻訊號的常規是此程序的一個例題。

如何將常規移開

輸入與輸出常規可利用 IN#0 與 PR#0從傳送口移開。或者在一個程式中可利用JSR SETKBD（位址 $FE89）模擬 IN#0，與JSR SETVID（位址 $FE93）模擬 PR#0。

DOS使用者注意事項

假如你使用的是磁碟作業系統（ DOS ），則當你將你的常規加入或移開時要遵守一些特別的規則。 DOS 平常使

用它自己的輸入與輸出傳送口。因此，當你改變傳送口時，你必須呼叫一個DOS常規通知DOS那個傳送口已經被改變了。DOS便會將本身接到那傳送口，但它就會使用你的常規，而不使用標準輸入/輸出常規。做此件工作的常規在位址$3EA。

範 例

在圖4的範例程式將從輸入傳送口加入或移掉一個常規。

要加入你的常規可從監督系統打入300G。要移掉你的常規則打入30CG。

```
300:  LDA  #low address of routine
302:  STA  $38   ;Store it in KSWL
304:  LDA  #high address byte of routine
306:  STA  $39   ;Store it in KSWH
308:  JSR  $3EA ;Reconnect DOS
30B:  RTS
30C:  JSR  $FE89;JSR SETKBD to simulate IN#0
30F:  JSR  $3EA ;Reconnect DOS
312:  RTS
```

圖 4

使用輸入傳送口的範例

這裡有三個字元是APPLE II 能接受的，但是不能從標準鍵盤打入。分別是“ / ”，“ [”，與“ — ”。解決的一種方法是修改鍵盤的硬體結構。另一種方法就是使用輸入傳送口，將沒有用到的控制字元轉換成這些字元。第一個程式將轉換下列字元：

CONTROL K → [
CONTROL L → /
CONTROL 0 → —

如何使用此程式

將程式利用BLOAD 命令存入 $300 開始的位址。注意，此程式是爲DOS 使用者寫的。假如你不是使用DOS，則利用JMP $3EA取代RTS 指令。要加入常規，則從監督系統打入 303G，或從BASIC打入CALL 771 。要移掉常規，則從監督系統打入 300G，或者從BASIC打入CALL 768。

第二個程式使用輸出傳送口將控制字元轉換成反相視頻字元。除了CONTROL M 控制字元外，它是一個Return訊

號，所有的控制字元都有轉換。

使用傳送口的一些重要位址

名稱	位址	說　　明
COUT1	$FDF0	監督系統字元輸出常規
CSWL	$36	輸出常規位址的低位元組
CSWH	$37	輸出常規位址的高位元組
KEYIN	$FD1B	監督系統輸入常規
KSWL	$38	輸入常規位址的低位元組
KSWH	$39	輸入常規位址的高位元組
MVSW	$3EA	連接 DOS 傳送口的常規
SETKBD	$FE89	模擬一個 IN#0
SETVID	$FE93	模擬一個 PR#0

```
0800        ;*************************
0800        ;*                       *
0800        ;*   HOW TO USE HOOKS    *
0800        ;*   RICHARD WILLIAMS    *
0800        ;*                       *
0800        ;*       NEWKEYS         *
0800        ;*                       *
0800        ;*  COPYRIGHT (C) 1981   *
0800        ;*   MICRO INK, INC.     *
0800        ;* CHELMSFORD, MA 01824  *
0800        ;*  ALL RIGHTS RESERVED  *
0800        ;*                       *
0800        ;*************************
0800        ;
0800        ;
0800        ;
0800        BKSLSH EPZ 220              ;ASCII BACKLASH
0800        CTRLK  EPZ 139              ;ASCII CONTROL K
0800        CTRLL  EPZ 140              ;ASCII CONTROL L
0800        CTRLO  EPZ 143              ;ASCII CONTROL O
```

```
0800            KSWL   EPZ $38             ;INPUT HOOK ADDRESS
0800            KSWH   EPZ $39
0800            RTBRKT EPZ 219             ;ASCII RIGHT BRACKET
0800            UNDSCR EPZ 223             ;ASCII UNDERSCORE
0800            ;
0800            ;
0800            KEYIN  EQU $FD1B           ;MONITOR'S INPUT HANDLER
0800            MVSW   EQU $3EA            ;ROUTINE TO RECONNECT DOS
0800            SETKBD EQU $FE89           ;SIMULATES IN#0
0800            ;
0800            ;
0800            ;------NEXT OBJECT FILE NAME IS NEWKEYS.OBJO
0800            ;
0800            ;
0800            ;
0300                   ORG $300
0300                   OBJ $800
0300            ;
0300            ;
0300            ;
0300 4C0F03            JMP UNHOOK          ;JUMP TO DISCONNECT ROUTINE
0303            ;
0303            ;*** THIS PART ATTACHES OUR ROUTINE INTO THE INPUT HOOK
0303            ;
0303 A916       ATTACH LDA #KEYCHK         ;A=LOW BYTE OF ADDRESS
0305 8538              STA KSWL
0307 A903              LDA /KEYCHK         ;GET HI BYTE
0309 8539              STA KSWH
030B 20EA03            JSR MVSW            ;GO TO IT
030E 60                RTS
030F            ;
030F            ;*** THIS PART UNHOOKS THE ROUTINE
030F            ;
030F 2089FE     UNHOOK JSR SETKBD          ;DO A IN#0
0312 20EA03            JSR MVSW
0315 60                RTS
0316            ;
0316            ;*** THIS IS THE ROUTINE
0316            ;
0316 201BFD     KEYCHK JSR KEYIN           ;GET THE KEY
0319 C98B              CMP #CTRLK          ;CONTROL K?
031B D003              BNE NOTK
031D A9DB              LDA #RTBRKT         ;MAKE IT A BRACKET
031F 60                RTS
0320 C98C       NOTK   CMP #CTRLL          ;CONTROL L?
0322 D003              BNE NOTL
0324 A9DC              LDA #BKSLSH         ;MAKE IT A BACKLASH
0326 60                RTS
0327 C98F       NOTL   CMP #CTRLO          ;CONTROL O?
0329 D002              BNE CHKDNE
032B A9DF              LDA #UNDSCR
032D 60         CHKDNE RTS
                       END

          *************************
          *                       *
          * SYMBOL TABLE -- V 1.5 *
          *                       *
          *************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

BKSLSH 00DC  CTRLK  008B  CTRLL  008C  CTRLO  008F  KSWL   0038  KSWH   0039
RTBRKT 00DB  UNDSCR 00DF
```

```
** ABSOLUTE VARABLES/LABELS

KEYIN  FD1B  MVSW   03EA  SETKBD FE89  ATTACH 0303
UNHOOK 030F  KEYCHK 0316  NOTK   0320  NOTL   0327  CHKDNE 032D

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:009A
```

```
0800           ;**************************
0800           ;*                        *
0800           ;*    HOW TO USE HOOKS    *
0800           ;*    RICHARD WILLIAMS    *
0800           ;*                        *
0800           ;*        CONVERT         *
0800           ;*                        *
0800           ;*   COPYRIGHT(C) 1981    *
0800           ;*     MICRO INK, INC.    *
0800           ;* CHELSMFORD, MA 01824   *
0800           ;*   ALL RIGHTS RESERVED  *
0800           ;*                        *
0800           ;**************************
0800           ;
0800           ;
0800           ;
0800           CSWH   EPZ $37            ;OUTPUT HOOK HIGH BYTE
0800           CSWL   EPZ $36            ;OUTPUT HOOK LOW ORDER BYTE
0800           CTRLM  EPZ $8D            ;CONTROL M
0800           MASK   EPZ $3F            ;MASK TO CONVERT TO INVERSE
0800           NULL   EPZ $80            ;NULL CHARACTER
0800           SPACE  EPZ $A0            ;SPACE CHARACTER
0800           ;
0800           ;
0800           ;
0800           COUT1  EQU $FDF0          ;CHARACTER OUTPUT ROUTINE
0800           MVSW   EQU $3EA           ;RECONNECTS DOS
0800           SETVID EQU $FE93          ;PERFORMS PR#0
0800           ;
0800           ;
0300                  ORG $300
0300                  OBJ $800
0300           ;
0300           ;
0300 4C0F03           JMP UNHOOK
0303           ;
0303           ;*** ROUTINE TO CONNECT ROUTINE INTO HOOK
0303           ;
0303 A916             LDA #CONVRT        ;GET LOW BYTE OF ADDRESS
0305 8536             STA CSWL
0307 A903             LDA /CONVRT        ;GET HIGH BYTE
0309 8537             STA CSWH
030B 20EA03           JSR MVSW
030E 60               RTS
030F           ;
```

```
030F              ;*** THIS UNHOOKS THE ROUTINE
030F              ;
030F 2093FE       UNHOOK JSR SETVID          ;SIMULATE PR#0
0312 20EA03              JSR MVSW            ;RECONNECT DOS
0315 60                  RTS
0316              ;
0316              ;*** THIS IS THE CONVERSION ROUTINE
0316              ;
0316 C980         CONVRT CMP #NULL           ;<NULL CHARACTER
0318 900A                BCC GOOUT
031A C9A0                CMP #SPACE          ;>=SPACE CHARACTER
031C B006                BCS GOOUT
031E C98D                CMP #CTRLM          ;RETURN CHAR?
0320 F002                BEQ GOOUT
0322 293F                AND #MASK           ;CONVERT TO INVERSE
0324 4CF0FD       GOOUT  JMP COUT1
                       END

***** END OF ASSEMBLY


          **************************
          *                        *
          * SYMBOL TABLE -- V 1.5  *
          *                        *
          **************************

LABEL. LOC.   LABEL. LOC.   LABEL. LOC.

** ZERO PAGE VARIABLES:

CSWH   0037  CSWL   0036  CTRLM  008D  MASK   003F  NULL   0080  SPACE  00A0


** ABSOLUTE VARABLES/LABELS

COUT1  FDF0  MVSW   03EA  SETVID FE93  UNHOOK 030F  CONVRT 0316  GOOUT  0324


SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:0072
```

BROWN和WHITE與所有的彩色

作者：Richard F. Suitor

視頻圖形的記憶體緩衝區是APPLE彩色能力的中樞地帶。本文將討論APPLE的彩色視頻輸出，強調彩色產生理論與彩色和螢幕記憶體位置之間的覆蓋關係。本文所探討的內容可用來產生一些隨機彩色顯示，這可以進一步的探討APPLE的圖形理論。

你的APPLE的彩色

你螢幕上的彩色是由你的彩色TV的視頻訊號（video signal）控制。大部份的訊號都含有圖形的明暗度資訊——利用這些訊號可產生黑白的圖形。在這訊號中再添加彩色載波，即一個3.58MHz的訊號，能夠含有彩色的資訊。訊號愈強，則圖形所涵蓋的地區色彩也就愈多。色彩（藍，綠，橙，等等。）是由彩色訊號的相位（phase）決定的。在每一掃瞄行的開始，參考時序信號（timing signal）使

得和"標準"彩色訊號同步。在一個3.85MHz的週期內，圖形彩色訊號升高時和標準彩色訊號升高時比較以決定色彩。當標準彩色訊號升高時，彩色訊號也升高，則表示爲橙色。若在此期間內彩色訊號下降，則表示爲藍色。當標準彩色訊號由高到低或由低到高時，而彩色訊號在高準位時，則表示爲紫色與綠色。（這只是強度問題。攝影室光線問題，輸送線，與使用者的天線都有影響。所以最後都要使用色彩控制或色濃度控制。）

彩色訊號對標準彩色訊號的時序關係表示爲"相位角度"，以角度的徑或度數表示，可以從0°到360°。相位角度相當於在彩色圖中的位置，橙色在頂端、藍色在底端，如圖1所示。

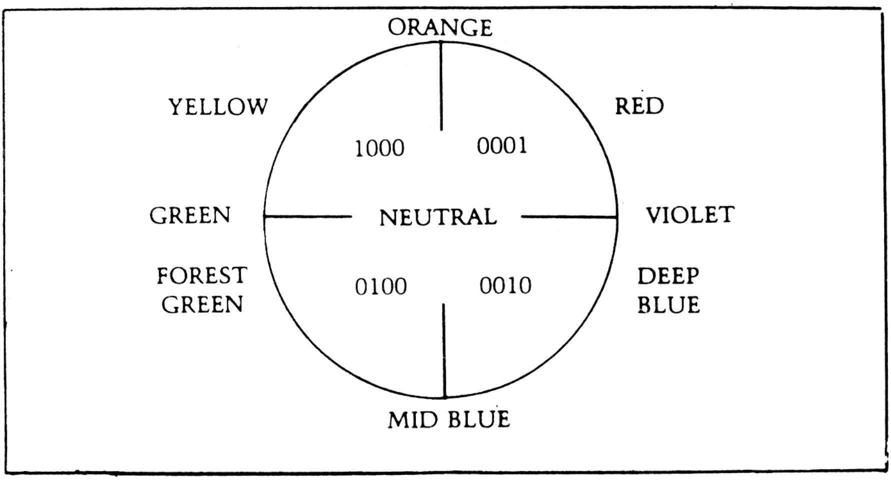


圖1　顯示色彩對色彩號碼位元位置關係的彩色圓

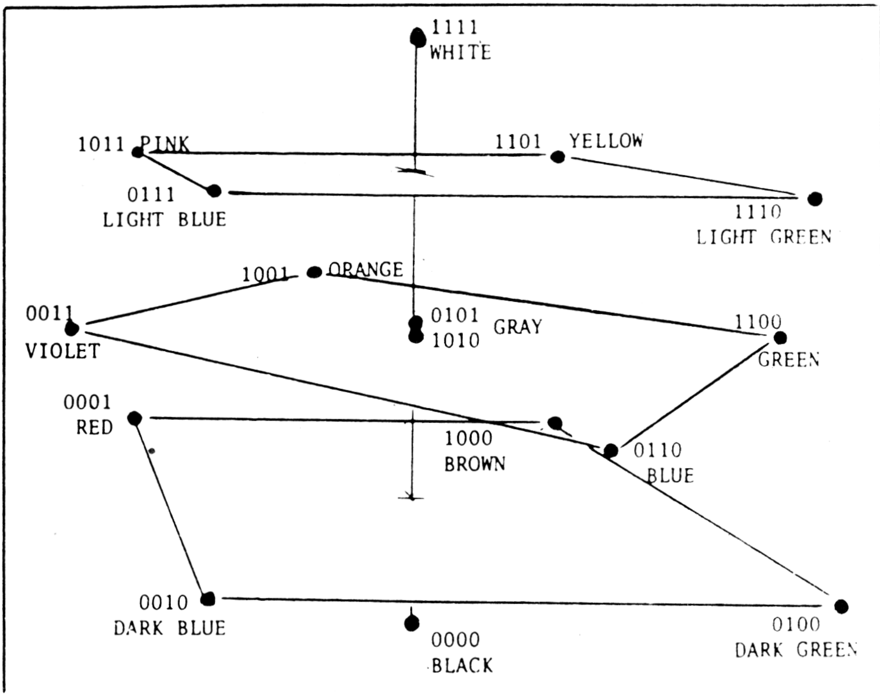
圓的周圍分別代表不同的色彩。從中心點的半徑距離代表色彩的飽和度。前者是由色濃度控制（tint control），後者由色彩控制（Color control）。能在彩色 TV 上顯示的色彩一定是在此圓內的相對一點。角度的位置是以 3.58 MHz 彩色載波訊號的相位來分；以中心點爲準的半徑就是彩色載波的振幅。

APPLE 的彩色編號可以用此彩色圓的色彩號碼二元表示法表示。最低位元相當於紅色（＃1）。第二個位元相當於深藍色（＃2），第三個位元相當於深綠色（＃4），最高位元相當於棕色（＃8）。每個位元各佔¼個彩色圓，並且分別對應其色彩，如圖 1 所示。色彩的明亮度相當於所佔¼圓的數目，且色彩相當於所有選定部份的平衡點。黑色，＃0，沒有設立任何位元，不佔圓的任何部，所以沒有亮度，白色，#15，4 個位元全部設立，佔滿整個圓，所以是最亮的，並且平衡點是在中間。橙色，＃9，或 1001，佔有圓的上半部，平衡點是在中心點和橙色之間。#5，或 0101，分別佔有二個分開的¼圓，一個是紅色，另一個是綠色。因爲它是互相對稱，所以平衡點在中間。它表示灰色，且明亮度剛好在中間，#10 也和 #5 一樣。#14，佔有除了紅色部份以外的圓，它的明亮度較強，平衡點在接近深綠色的一條線上，它表示青綠色。

因此，APPLE 的彩色就用這些數字來表示。你可以想成它們是紅色，深藍色，深綠色，與棕色的加法組合，將二個色彩加起來就是將二個色彩的數字 OR 起來。減法組合就

是將二個較亮的色彩的數字 AND 起來，減法組合的色彩計有粉紅色，黃色，淡綠色，與淡藍色。在數字中所設立的位元愈多，色彩就愈明亮；設立的位元愈少，就愈暗。數字 5 和 10 沒有 3.58 MHz 的成分存在，所以產生的色彩不亮也不暗，剛好適中，在 5 和 10 之間的數字，不是二個 1 連在一起就是兩個 0 連在一起。試試下列程式，它將只會顯示灰色：

```
 10 GR
 20 FOR I=0 TO 9
 30 COLOR =5
 40 HLIN 0,39 AT 2*I
 50 VLIN 20,39 AT 2*I
 60 VLIN 20,39 AT 2*I+21
 70 COLOR=10
 80 HLIN 0,39 AT 2*I+1
 90 VLIN 20,39 AT 2*I+1
100 VLIN 20,39 AT 2*I+20
110 NEXT I
120 RETURN
```



APPLE II 彩色的色彩空間位置

圖 2　每一個水平面形成一個不同明亮度的彩色圓

在顯示圖形的上半部是由HLIN畫出來的，色彩號碼爲5和10。下半部是VLIN 畫出來的，色彩號碼也是5與10。你看到什麼了？色彩號碼的位元直接放在視頻訊號內，4個位元佔一個彩色載波週期。當二個1的位元相連在一起時，會形成一個亮的波帶（ band ）。當二個0的位元相連在一起時，會形成一個暗的波帶。

現在看看3.58MHz週期有多大。APPLE的正文(text)是由5×7的點矩陣組成的，是一種普遍的字元產生法。這些點相當於高解析度顯示記憶體中的位元。一點是½個3.58MHz週期，並且相當於一個紫色(#3)或綠色(#12)彩色訊號。這就是爲什麼正文在彩色電視機上是比較亮的色彩，而在高解析度的顯示下有二種色彩，即綠色和紫色。

[註：並不是每一個高解析度的點都能繪上特定的色彩。以綠色來說，只能畫出一半，另一半可能是紫色，這就是爲什麼在高解析度下畫一點或垂直線，有時看起來好像沒有繪出什麼。重複畫二次便可解決此問題。]

彩色 TV已經進一步的在爲顯示圖形而設計。黑色和白色視頻訊號大約限制在4MHz，且有一些受像機(sets)會漏掉顯示頻率反應，所以彩色訊號便不會顯現。像這種受像機不能把點解析得很好，所以所產生的正文有點模糊。有些彩色受像機經過調整後，能使得受像機忽略彩色訊號。因爲彩色訊號的處理牽涉到訊號的加與減，避免這個便可增進正文的解析度。減少對比與明亮度也可改進正文資料。

彩色 TV的設計想要從圖形中移掉彩色載波(在順利的提供彩色後)，但是你可能在每一個彩色方塊中看到3或4條細微的垂直線。在全白，全黑或全灰時，它們都不應該出現的(除非在一個高解析度監督程式)。

黃褐色是介於棕色和白色之間

此節將說明APPLE色彩在彩色空間上的關係。我想各位都是Scientific American雜誌Martin Gardner “Mathematical Games ”專欄的忠實讀者。我鄭重推荐它是很好的專欄。

有一篇專欄討論到不同性質的隨機變化所產生美的特性。簡單的說，三種性質為：

WHITE	每個分離的元素都是任意取的，並且和其它元素沒有任何關係。這就稱做“ White ”，因為頻率光譜顯示出所有的頻率都相等，這是白色光的定性描述。
BROWN	每個分離的元素是前一個元素加上一個任意取的偏差值。這就稱做“ brown ”，因為布朗寧運動（ Brownian motion ）就是一個例子。
1/F	它的頻率光譜介乎“ white ”和“ brown ”之間。

此篇專欄證明1/f變化比“ white ”（不夠統一），或“ brown ”（不夠變化）較好較美。在此提供一個方法在1/f隨機變化下產生元素。簡單的說，每個元素是N個項的總和。一項是對每一個元素任意選擇。下一項是對其它每一個元素任意選擇。下一項是對每4個元素任意選擇，等等。

利用APPLE，你可以利用這些概念做一些實驗。在專欄中並沒有提到彩色的問題。此節將把這些概念應用到APPLE圖形顯示上。

我們都知道在APPLE圖形顯示中，“ white ”雜訊是什麼樣子。我們可以試一下，在螢幕上任選一點，再任選一個任意色彩，然後重複繪在螢幕上。例如：

```
10  GR
20  X = RND(40)
30  Y = RND(40)
40  COLOR = RND(16)
50  PLOT X,Y
60  GOTO 20
```

所顯示的結果會很耀眼，這就是“ white ”型式的隨機圖形顯示。在螢幕上，某個小方塊的色彩和它相鄰的小方塊色彩沒有任何關係，並且在繪出某一小方塊後，並不知道下一個小方塊要繪在那裏。

我利用下列幾點說明“ 1/f ”的概念：

1. X和Y分別是3個數字的和，一個是每次繪的時候任意選一數字，另一個每 20 次選一個，第三個是每200次選一個。

2. 設一個彩色號碼表（在程式中是DIM(16)），所以當彩色號碼接近時，相對的它們的色彩也會接近。在選擇彩色號碼時要符合下列限制：

   a. 相鄰的號碼要從圖2相鄰的平面中選擇。

b. 二個相鄰號碼其在彩色平面上的角度變化不能大於 45° 。

3. 對於 20 次繪出的點，其色彩號碼都一樣， 所以要加上 −2 到 +2 之間的任意值改變。這是“ brown ”雜訊產生的概念。然而大部份的圖形顯示中都含有一些已經產生過的彩色碎點，因此和現時所繪的點較沒有關係。

4. 每“ 繪一次 ”事實上是對一些主軸對稱地繪 8 點。這和 1/f 沒有關係，不過可得出萬花筒的效果。各位如果不喜歡的話，可以改變行號 100 。各位也可以使 X 和 Y 爲更多項的和，可以在較大的區間內選擇第 4 項和第 5 項。

利用 paddle和按鈕控制速度與重新設定畫面。假如你的 paddle 沒有接上，利用 0 取代 PDL (0) 。

```
  1 REM  *************************
  2 REM  *                       *
  3 REM  * BROWN,WHITE,COLOURED  *
  4 REM  *    RICHARD SUITOR     *
  5 REM  *                       *
  6 REM  *     BROWN/WHITE       *
  7 REM  *                       *
  8 REM  *                       *
  9 REM  *  COPYRIGHT (C) 1981   *
 10 REM  *    MICRO INK, INC.    *
 11 REM  * CHELMSFORD, MA 01824  *
 12 REM  *  ALL RIGHTS RESERVED  *
 13 REM  *                       *
 14 REM  *************************
 20 DIM A(16):A(1)=0:A(2)=2:A(3)=6:A(4)=7:A(5)=3:A(6)=1:A(7)=5:A(8)=11
 22 A(9)=9:A(10)=8:A(11)=10:A(12)=13:A(13)=15:A(14)=14:A(15)=12:A(16)=4

 40 GOTO 3000
100 PLOT X,Y: PLOT 38-X,Y: PLOT X,38-Y: PLOT 38-X,38-Y: PLOT Y,X: PLOT
    38-Y,38-X: PLOT Y,38-X: PLOT 38-Y,X
110 RETURN
120 Z=16
125 L= RND (5)-2
130 U= RND (9):V= RND (9)
```

```
 147 FOR B=1 TO 10
 150 R=U+ RND (9):S=V+ RND (9)
 155 IF PEEK (-16286)>127 THEN GR
 160 K=K+L: IF K>16 THEN K=K-Z
 165 IF K<0 THEN K=K+Z
 170 COLOR=A(K)
 180 Q=( PDL (0)/2) ^ 2
 190 FOR I=-Q TO Q: IF PEEK (-16287)>127 THEN 200: NEXT I
 200 FOR I=1 TO 20
 210 X=R+ RND (6):Y=S+ RND (6): GOSUB 100: NEXT I
 220 NEXT B
 230 GOTO 120
1010 K=1:L=5
1020 Z=16
2000 GOTO 120
3000 GR : CALL -936
3010 PRINT "PADDLE 0 CONTROLS PATTERN SPEED"
3020 PRINT "USE BUTTON 0 TO GO AT ONCE TO HI SPEED"
3030 PRINT "HOLD BUTTON 1 TO CLEAR SCREEN"
3040 GOTO 1010
9000 END
```

```
DISK VOLUME 002

*A 005 MICRO ON THE APPLE 2
*B 004 BREAKER
*B 002 STEP-TRACE
*B 002 STEP-TRACE.800
*B 002 TRACER
*B 003 PACK-LOAD
*B 002 MEAN-14
*B 002 SCREEN WRITE
*I 007 SCROLLER
*B 002 SCROLL
*B 005 PAGE LIST
*B 002 PAGE
*B 002 HEX PRINTER
*B 002 COM-VAR-I
*B 002 COM-VAR-A
*B 002 PRINT USING
*A 007 SEARCH
*B 002 STRING SEARCH
*A 008 MATRIX DEMO
*B 008 MATRICES
*A 009 AMPERSORT DEMO
*B 005 AMPERSORT
*I 009 TRACE LIST
*B 003 TRACE INTERRUPT
*I 003 TRACE TEST
*A 015 FNPLOTTER
*I 011 COMPRESS
*B 010 PICT COMP
*B 034 LADY BE GOOD
*I 016 LIFESAVER
*B 003 LIFE
*I 004 APPLAYER MENU
*B 010 APPLAYER
*I 004 BATTLE SOUNDS
*I 022 GALACTI-CUBE
*A 007 COLOR GUN
*B 003 CASSCS
*I 005 DIRECTORY
*A 005 BASIC TRANSFER
*A 010 BINARY TRANSFER
*A 014 THERMOMETER
*B 003 SYM-KIM
*B 002 ERROR
*A 007 FLOATING POINT
*B 002 NEWKEYS
*B 002 CONVERT
*I 006 BROWN/WHITE
```

PUBLISHED & PRINTED IN HONG KONG H.K.$ 25.00

APPLE 軟體 硬體 應用程式

廖敏芳譯

博文出版社出版